AppleScript for Programers

Essential AppleScript Tools & Tips

Session 138

These are confidential sessions—please refrain from filming or taking pictures.

Application Technologies

AppleScript for Programers

Learning to embrace the laid-back language

Sal Soghoian
AppleScript Product Manager
sal@apple.com

http://developer.apple.com/wwdc2005/

Who is this session for?

- Experience in programming standard languages such as Java, Objective-C, C++, Python, perl, etc.
- An interest in learning AppleScript

What You'll Learn

- What is AppleScript?
- AppleScript Language Overview
  - targeting scriptable applications
  - referring to scriptable items
  - finding items
  - constructs: repeat loops, conditionals, error handlers
- AppleScript Tools
  - Script Editor and Script Menu

What is AppleScript?

http://developer.apple.com/wwdc2005/

What is AppleScript?

A high-level, English-like language used to query and control scriptable applications

Designed to be used by “non-programers”

Well-integrated into Mac OS X:

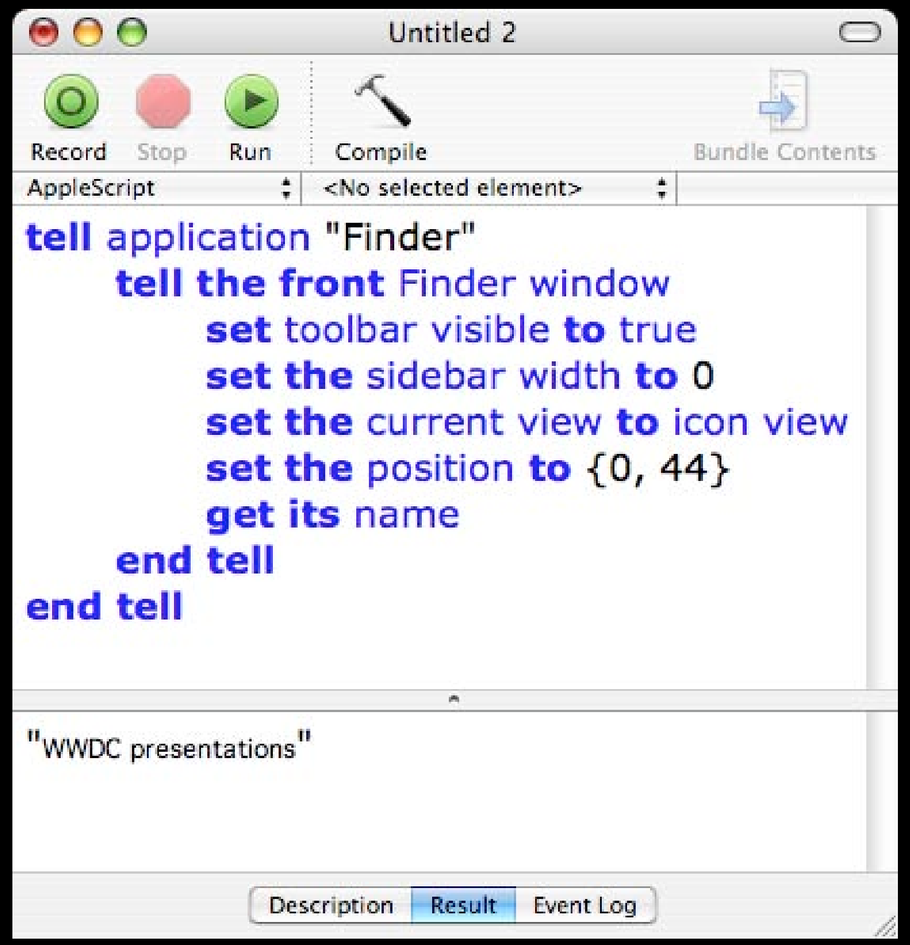
- Scriptable applications: iPhoto, iTunes, iDVD, iCal, etc.
- Scriptable access to frameworks: Shell, Webservices, Data Events, Image Events, XML parsing, Plist parsing
- AppleScript Studio
- Automator

What is AppleScript?

80% of Apple's top-tier customers use AppleScript

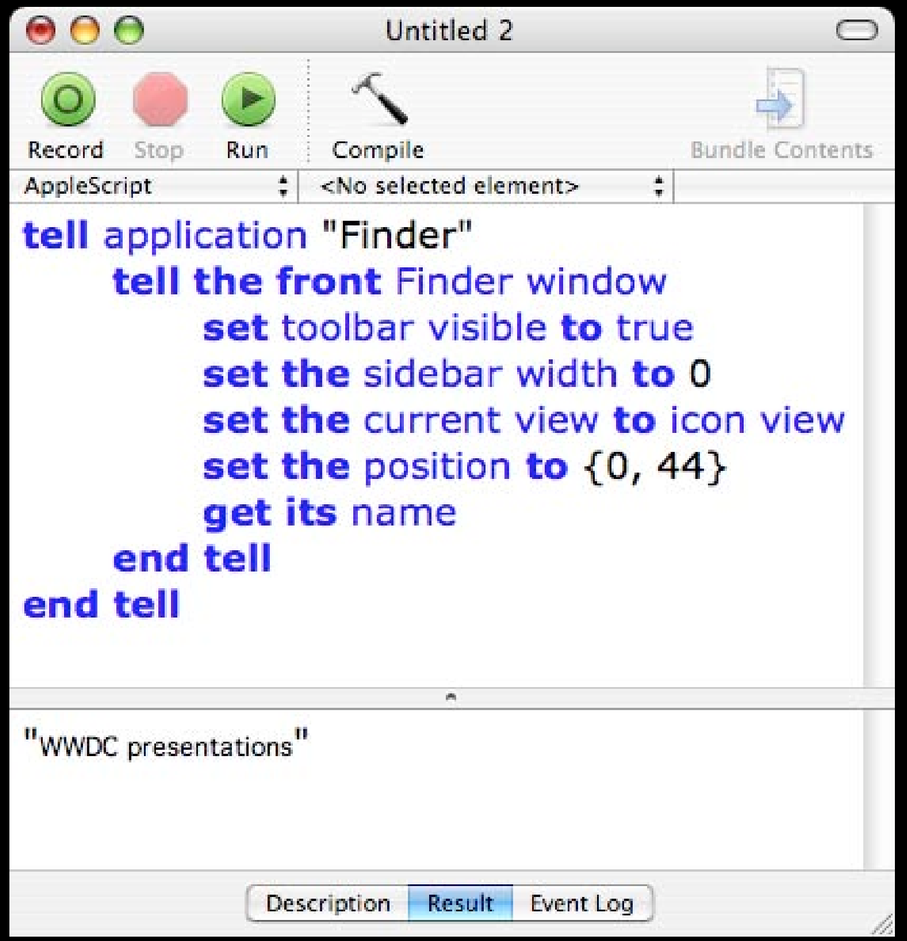
How does it work?

- AppleScript “scripts” are composed in a script editing application like Script Debugger or the Script Editor.



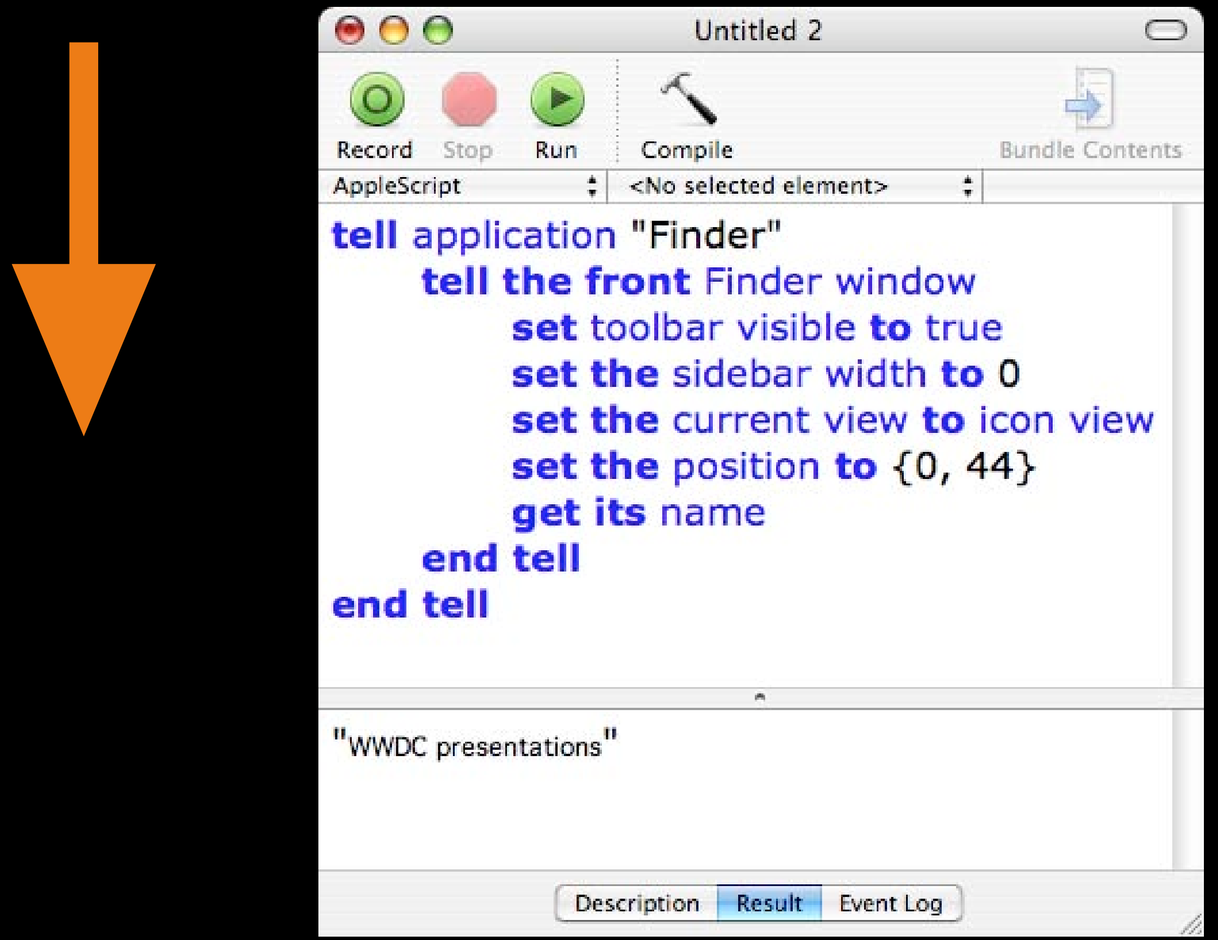
How does it work?

- Scripts are read from left to right…



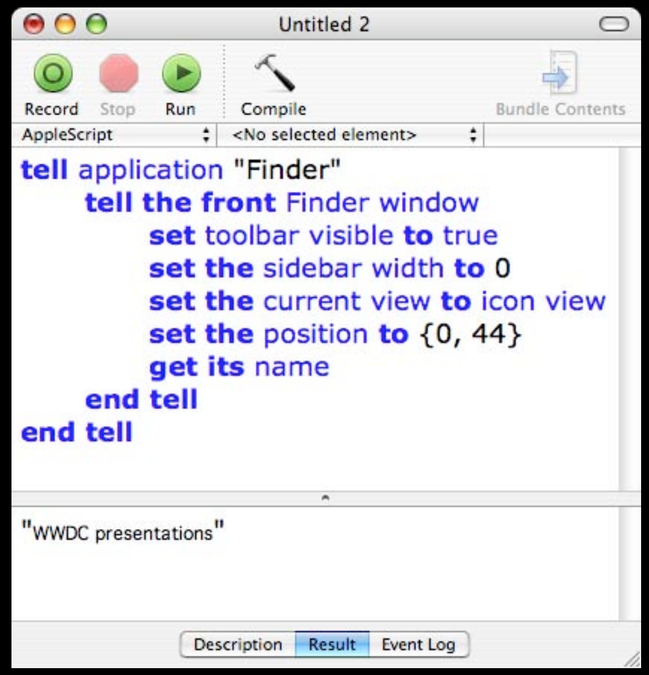
How does it work?

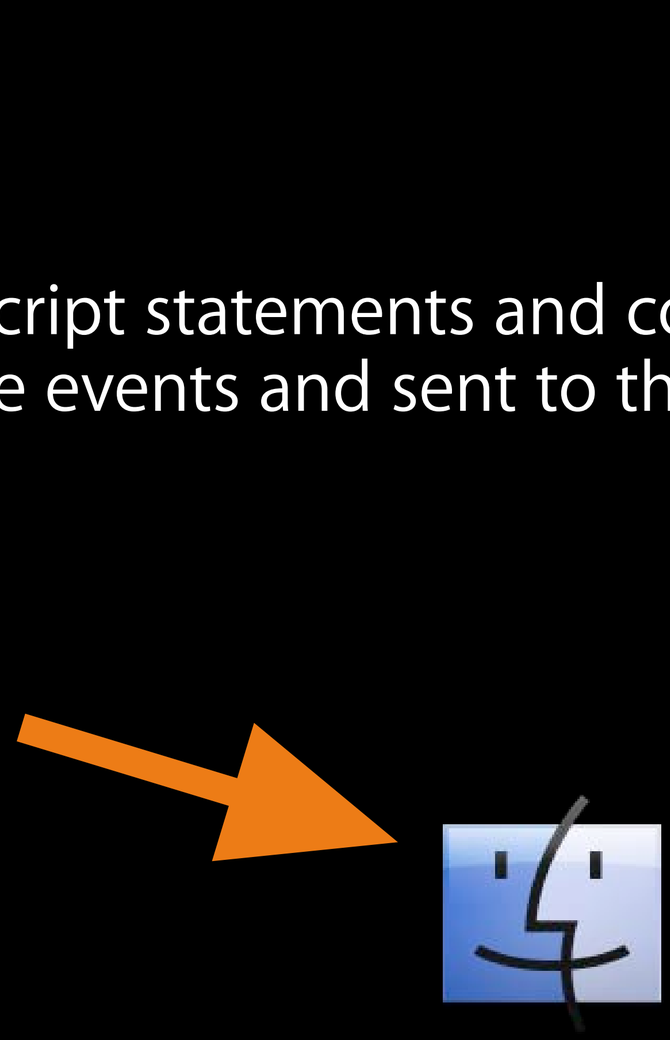
- and from the top to the bottom.



How does it work?

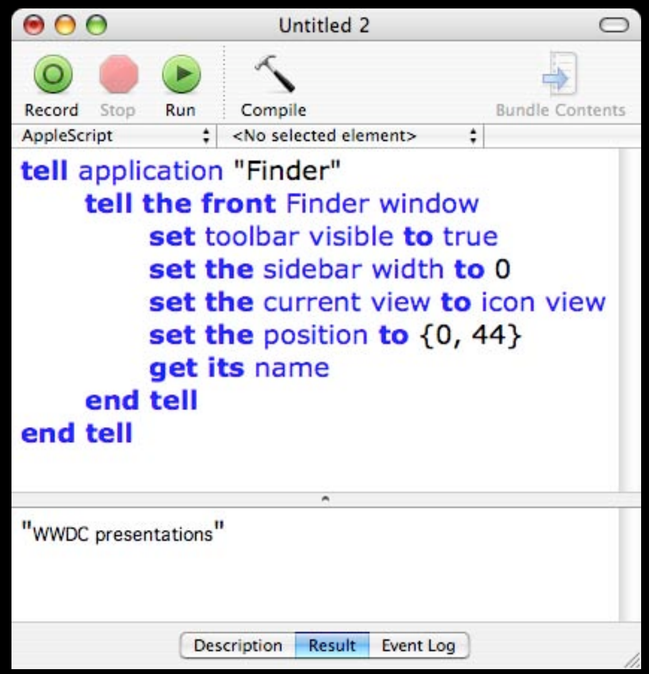
- When executed, AppleScript statements and commands are converted into Apple events and sent to the targeted applications.

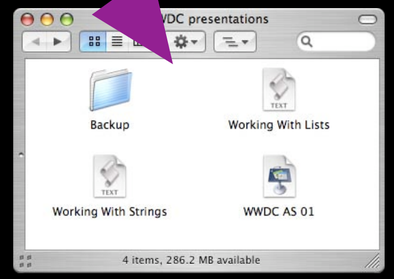




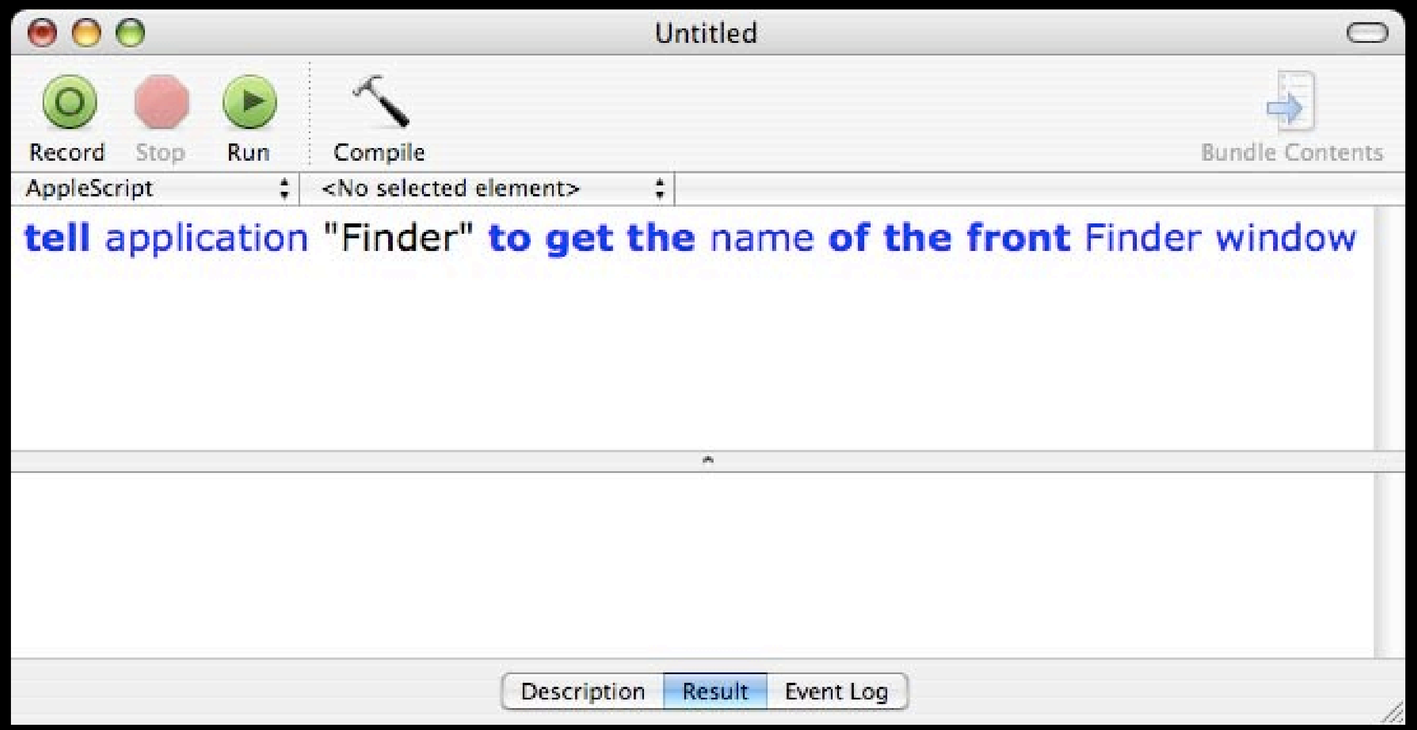
How does it work?

- Commands/queries are executed by the targeted application and the results are returned to the script unless otherwise indicated.

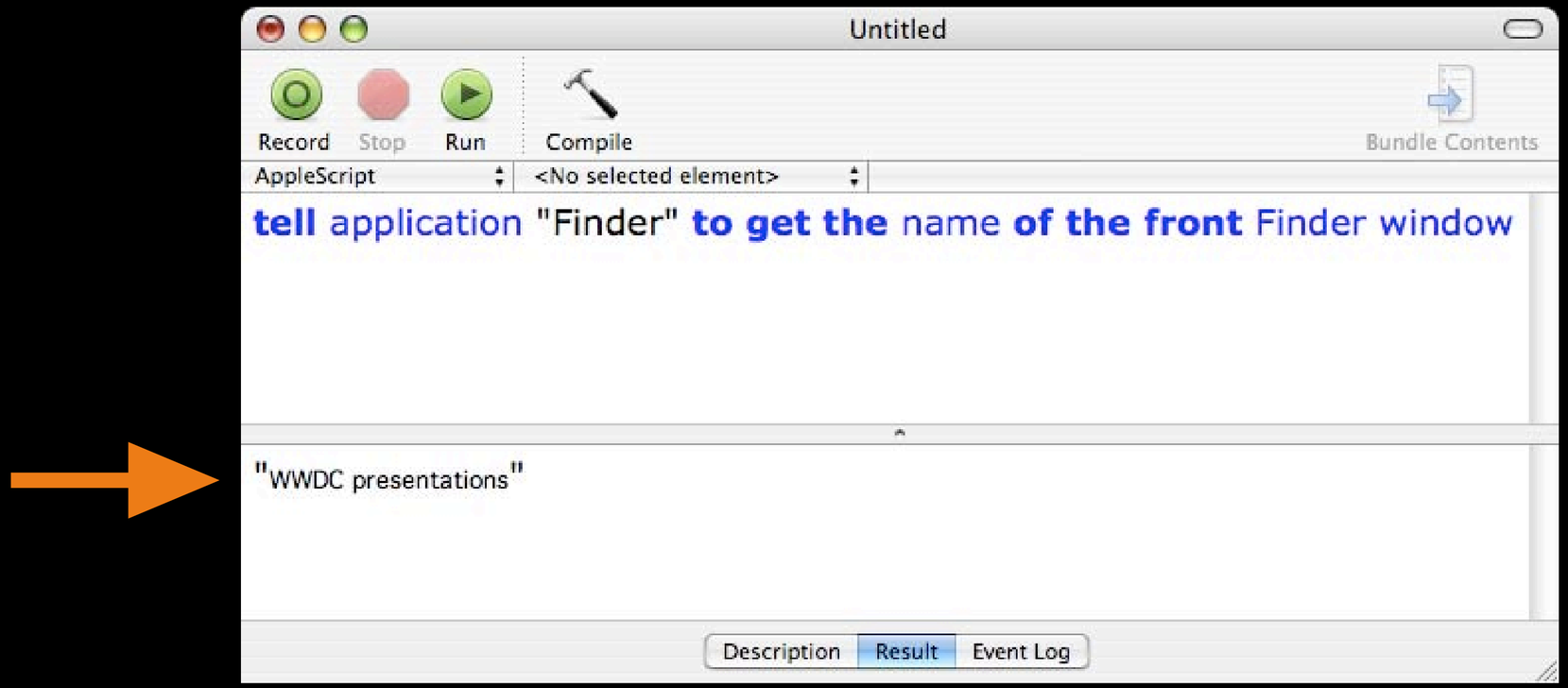




How does it work?

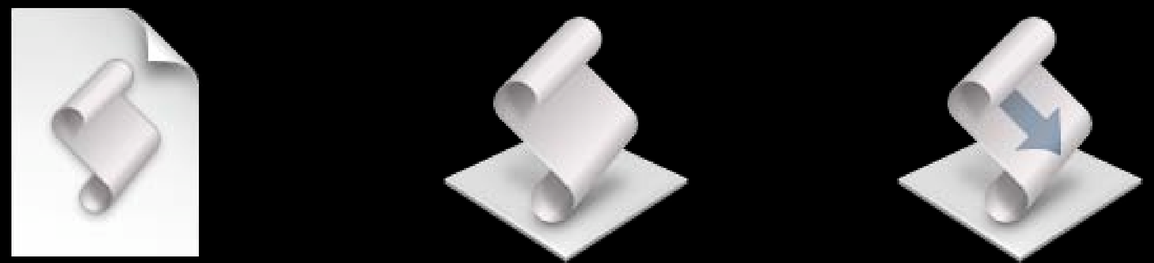


How does it work?



How does it work?

- Scripts can be saved as documents that executed from a script menu, or as self-running applications.



AppleScript has an “English-like” Syntax

- “Move the green car before the red car.”
- “Get the name of every student whose graduation comes after June 1st.”
- “If your first name begins with “B” then stand in the line on the left, otherwise, stand in the line on the right.”

Syntax Comparison:

```
app('Finder').home.folders.name.get()
```

Syntax Comparison:

```
app('Finder').home.folders.name.get()
```

Syntax Comparison:

```
app('Finder').home.folders.name.get()
```

Syntax Comparison:

```
app('Finder').home.folders.name.get()
```

Syntax Comparison:

```
app('Finder').home.folders.name.get()
```

Syntax Comparison:

```
app('Finder').home.folders.name.get()
```

Syntax Comparison:

```
app('Finder').home.folders.name.get()
```

```
tell app "Finder" to get the name of every folder of home
```

Syntax Comparison:

```
app('TextEdit').documents[1].words.count()
```

```
tell app "TextEdit" to count the words of the first document
```

Syntax Comparison:

```
app('TextEdit').documents[1].paragraphs.test(its == '\n').delete()
```

```
tell app "TextEdit" to delete ¬
  (every paragraph of the front document where it is "\n")
```

AppleScript Syntax

```
set property of object to value

set the color of the current box to "Red"

set the name of the third track of the first playlist to "Fred"

set the label index of every folder of home to 3
```

AppleScript Syntax

```
tell the object to set its property to this_value

tell the current box to set its color to "Red"

tell the third track of the first playlist to set its name to "Fred"

tell every folder of home to set its label index to 3
```

AppleScript Syntax

```
tell the object to set its property to this_value

tell the current box to set its color to "Red"

tell the third track of the first playlist to set its name to "Fred"

tell every folder of home to set its label index to 3
```

The Syntax of AppleScript is…

- Similar to English
- “Says what it does.” (most of the time)
- Flexible - often more than one way to “say” the same thing
- Verbose!

“Talking” to Applications

Tell Statements and Tell Blocks

http://developer.apple.com/wwdc2005/

Tell Statement

Targets a scriptable application

Single line beginning with the verb: tell

Contains two elements:

- a reference to the object to be scripted
- the action to be performed

Tell Statement

```
tell app "QuickTime Player" to tell the front movie to ¬
  tell the last track to get its duration
```

Tell Statement: reference to object

```
tell app "QuickTime Player" to tell the front movie to ¬
  tell the last track to get its duration
```

Tell Statement: action to perform

```
tell app "QuickTime Player" to tell the front movie to ¬
  tell the last track to get its duration
```

Tell Statement: syntax

```
tell app "QuickTime Player" to tell the front movie to ¬
  tell the last track to get its duration
```

Tell Statement: syntax

```
tell app "QuickTime Player" to tell the front movie to ¬
  tell the last track to get its duration

tell app "QuickTime Player" to get the duration of ¬
  the last track of the front movie
```

Tell Statement: reference to object

```
tell app "QuickTime Player" to tell the front movie to ¬
  tell the last track to get its duration

tell app "QuickTime Player" to get the duration of ¬
  the last track of the front movie
```

Tell Statement: action to perform

```
tell app "QuickTime Player" to tell the front movie to ¬
  tell the last track to get its duration

tell app "QuickTime Player" to get the duration of ¬
  the last track of the front movie
```

Tell Statement: syntax

```
tell app "QuickTime Player" to tell the front movie to ¬
  tell the last track to get its duration

tell app "QuickTime Player" to get the duration of ¬
  the last track of the front movie
```

Tell Statement: syntax

```
tell app "QuickTime Player" to tell the front movie to ¬
  tell the last track to get its duration

tell app "QuickTime Player" to get the duration of ¬
  the last track of the front movie
```

Tell Statement

```
tell application "iTunes" to delete (every track of playlist "My Party Mix" whose
     artist is "Bon Jovi")
```

Tell Statement

```
tell application "iTunes" to delete (every track of playlist "My Party Mix" whose
     artist is "Bon Jovi")
```

Tell Statement

```
tell application "iTunes" to delete (every track of playlist "My Party Mix" whose
     artist is "Bon Jovi")
```

Tell Statement

```
tell application "Finder" to duplicate (every document file of the entire
     contents of folder "Documents" of home whose name contains "Smith Project")
     to the disk named "Backup"
```

Tell Statement

```
tell application "Finder" to duplicate (every document file of the entire
     contents of folder "Documents" of home whose name contains "Smith Project")
     to the disk named "Backup"
```

Tell Statement

```
tell application "Finder" to duplicate (every document file of the entire
     contents of folder "Documents" of home whose name contains "Smith Project")
     to the disk named "Backup"
```

Tell Block

Target one or more actions at a scriptable object

Opening statement begins with the verb: tell

Ends with: end tell

Tell Block

```
tell application "Finder"
    set desktop shows removable media of the Finder preferences to false
    set desktop shows hard disks of the Finder preferences to false
    set desktop shows connected servers of the Finder preferences to false
    set folders open in new windows of the Finder preferences to false
    set new windows open in column view of the Finder preferences to false
    set all name extensions showing of the Finder preferences to true
    close every window
    open home
    set the current view of the front window to column view
    set the sidebar width of the front window to 0
    set the position of the front window to {0, 44}
    open folder "Documents" of home
    set the current view of the front window to icon view
    set the sidebar width of the front window to 0
    set the position of the front window to {72, 144}
end tell
```

Tell Block

```
tell application "Finder"
    set desktop shows removable media of the Finder preferences to false
    set desktop shows hard disks of the Finder preferences to false
    set desktop shows connected servers of the Finder preferences to false
    set folders open in new windows of the Finder preferences to false
    set new windows open in column view of the Finder preferences to false
    set all name extensions showing of the Finder preferences to true
    close every window
    open home
    set the current view of the front window to column view
    set the sidebar width of the front window to 0
    set the position of the front window to {0, 44}
    open folder "Documents" of home
    set the current view of the front window to icon view
    set the sidebar width of the front window to 0
    set the position of the front window to {72, 144}
end tell
```

Tell Block

```
tell application "Finder"
    set desktop shows removable media of the Finder preferences to false
    set desktop shows hard disks of the Finder preferences to false
    set desktop shows connected servers of the Finder preferences to false
    set folders open in new windows of the Finder preferences to false
    set new windows open in column view of the Finder preferences to false
    set all name extensions showing of the Finder preferences to true
    close every window
    open home
    set the current view of the front window to column view
    set the sidebar width of the front window to 0
    set the position of the front window to {0, 44}
    open folder "Documents" of home
    set the current view of the front window to icon view
    set the sidebar width of the front window to 0
    set the position of the front window to {72, 144}
end tell
```

Tell Block

```
tell application "Finder"
    set desktop shows removable media of the Finder preferences to false
    set desktop shows hard disks of the Finder preferences to false
    set desktop shows connected servers of the Finder preferences to false
    set folders open in new windows of the Finder preferences to false
    set new windows open in column view of the Finder preferences to false
    set all name extensions showing of the Finder preferences to true
    close every window
    open home
    set the current view of the front window to column view
    set the sidebar width of the front window to 0
    set the position of the front window to {0, 44}
    open folder "Documents" of home
    set the current view of the front window to icon view
    set the sidebar width of the front window to 0
    set the position of the front window to {72, 144}
end tell
```

Tell Block

```
tell application "Finder"
    set desktop shows removable media of the Finder preferences to false
    set desktop shows hard disks of the Finder preferences to false
    set desktop shows connected servers of the Finder preferences to false
    set folders open in new windows of the Finder preferences to false
    set new windows open in column view of the Finder preferences to false
    set all name extensions showing of the Finder preferences to true
    close every window
    open home
    set the current view of the front window to column view
    set the sidebar width of the front window to 0
    set the position of the front window to {0, 44}
    open folder "Documents" of home
    set the current view of the front window to icon view
    set the sidebar width of the front window to 0
    set the position of the front window to {72, 144}
end tell
```

Nested Tell Blocks

Tell blocks can contain other tell blocks

Better visual organization

Nested Tell Blocks

```
tell application "Finder"
    set desktop shows removable media of the Finder preferences to false
    set desktop shows hard disks of the Finder preferences to false
    set desktop shows connected servers of the Finder preferences to false
    set folders open in new windows of the Finder preferences to false
    set new windows open in column view of the Finder preferences to false
    set all name extensions showing of the Finder preferences to true
    close every window
    open home
    set the current view of the front window to column view
    set the sidebar width of the front window to 0
    set the position of the front window to {0, 44}
    open folder "Documents" of home
    set the current view of the front window to icon view
    set the sidebar width of the front window to 0
    set the position of the front window to {72, 144}
end tell
```

Nested Tell Blocks

```
tell application "Finder"
    set desktop shows removable media of the Finder preferences to false
    set desktop shows hard disks of the Finder preferences to false
    set desktop shows connected servers of the Finder preferences to false
    set folders open in new windows of the Finder preferences to false
    set new windows open in column view of the Finder preferences to false
    set all name extensions showing of the Finder preferences to true
    close every window
    open home
    set the current view of the front window to column view
    set the sidebar width of the front window to 0
    set the position of the front window to {0, 44}
    open folder "Documents" of home
    set the current view of the front window to icon view
    set the sidebar width of the front window to 0
    set the position of the front window to {72, 144}
end tell
```

Nested Tell Blocks

```
tell application "Finder"
    tell Finder preferences
        set desktop shows hard disks to false
        set desktop shows removable media to false
        set desktop shows connected servers to false
        set folders open in new windows to false
        set new windows open in column view to false
        set all name extensions showing to true
    end tell
    close every window
    open home
    set the current view of the front window to column view
    set the sidebar width of the front window to 0
    set the position of the front window to {0, 44}
    open folder "Documents" of home
    set the current view of the front window to icon view
    set the sidebar width of the front window to 0
    set the position of the front window to {72, 144}
end tell
```

Nested Tell Blocks

```
tell application "Finder"
    tell Finder preferences
        set desktop shows hard disks to false
        set desktop shows removable media to false
        set desktop shows connected servers to false
        set folders open in new windows to false
        set new windows open in column view to false
        set all name extensions showing to true
    end tell
    close every window
    open home
    set the current view of the front window to column view
    set the sidebar width of the front window to 0
    set the position of the front window to {0, 44}
    open folder "Documents" of home
    set the current view of the front window to icon view
    set the sidebar width of the front window to 0
    set the position of the front window to {72, 144}
end tell
```

Nested Tell Blocks

```
tell application "Finder"
    tell Finder preferences
        set desktop shows hard disks to false
        set desktop shows removable media to false
        set desktop shows connected servers to false
        set folders open in new windows to false
        set new windows open in column view to false
        set all name extensions showing to true
    end tell
    close every window
    open home
    set the current view of the front window to column view
    set the sidebar width of the front window to 0
    set the position of the front window to {0, 44}
    open folder "Documents" of home
    set the current view of the front window to icon view
    set the sidebar width of the front window to 0
    set the position of the front window to {72, 144}
end tell
```

Nested Tell Blocks

```
tell application "Finder"
    tell Finder preferences
        set desktop shows hard disks to false
        set desktop shows removable media to false
        set desktop shows connected servers to false
        set folders open in new windows to false
        set new windows open in column view to false
        set all name extensions showing to true
    end tell
    close every window
    open home
    set the current view of the front window to column view
    set the sidebar width of the front window to 0
    set the position of the front window to {0, 44}
    open folder "Documents" of home
    set the current view of the front window to icon view
    set the sidebar width of the front window to 0
    set the position of the front window to {72, 144}
end tell
```

Nested Tell Blocks

```
tell application "Finder"
    tell Finder preferences
        set desktop shows hard disks to false
        set desktop shows removable media to false
        set desktop shows connected servers to false
        set folders open in new windows to false
        set new windows open in column view to false
        set all name extensions showing to true
    end tell
    close every window
    open home
    set the current view of the front window to column view
    set the sidebar width of the front window to 0
    set the position of the front window to {0, 44}
    open folder "Documents" of home
    set the current view of the front window to icon view
    set the sidebar width of the front window to 0
    set the position of the front window to {72, 144}
end tell
```

Nested Tell Blocks

```
tell application "Finder"
    tell Finder preferences
        set desktop shows hard disks to false
        set desktop shows removable media to false
        set desktop shows connected servers to false
        set folders open in new windows to false
        set new windows open in column view to false
        set all name extensions showing to true
    end tell
    close every window
    open home
    tell the front window
        set the current view to column view
        set the sidebar width to 0
        set the position to {0, 44}
    end tell
    open folder "Documents" of home
    tell the front window
        set the current view to icon view
        set the sidebar width to 0
        set the position to {72, 144}
    end tell
end tell
```

Nested Tell Blocks

```
tell application "Finder"
    tell Finder preferences
        set desktop shows hard disks to false
        set desktop shows removable media to false
        set desktop shows connected servers to false
        set folders open in new windows to false
        set new windows open in column view to false
        set all name extensions showing to true
    end tell
    close every window
    open home
    tell the front window
        set the current view to column view
        set the sidebar width to 0
        set the position to {0, 44}
    end tell
    open folder "Documents" of home
    tell the front window
        set the current view to icon view
        set the sidebar width to 0
        set the position to {72, 144}
    end tell
end tell
```

Nested Tell Blocks

```
tell application "Finder"
    set desktop shows removable media of the Finder preferences to false
    set desktop shows hard disks of the Finder preferences to false
    set desktop shows connected servers of the Finder preferences to false
    set folders open in new windows of the Finder preferences to false
    set new windows open in column view of the Finder preferences to false
    set all name extensions showing of the Finder preferences to true
    close every window
    open home
    set the current view of the front window to column view
    set the sidebar width of the front window to 0
    set the position of the front window to {0, 44}
    open folder "Documents" of home
    set the current view of the front window to icon view
    set the sidebar width of the front window to 0
    set the position of the front window to {72, 144}
end tell
```

Nested Tell Blocks

```
tell application "Finder"
    tell Finder preferences
        set desktop shows hard disks to false
        set desktop shows removable media to false
        set desktop shows connected servers to false
        set folders open in new windows to false
        set new windows open in column view to false
        set all name extensions showing to true
    end tell
    close every window
    open home
    tell the front window
        set the current view to column view
        set the sidebar width to 0
        set the position to {0, 44}
    end tell
    open folder "Documents" of home
    tell the front window
        set the current view to icon view
        set the sidebar width to 0
        set the position to {72, 144}
    end tell
end tell
```

Nested Tell Blocks: Multi-level

```
tell application "Finder"
    tell home
        tell folder "Documents"
            tell folder "Smith Project"
                tell every item
                    set label index to 3
                    set comment to "Smith Project Final"
                end tell
            end tell
        end tell
    end tell
```

Nested Tell Blocks: Multi-level

```
tell application "Finder"
    tell home
        tell folder "Documents"
            tell folder "Smith Project"
                tell every item
                    set label index to 3
                    set comment to "Smith Project Final"
                end tell
            end tell
        end tell
    end tell

    tell application "Finder"
        tell every item of folder "Smith Project" of folder "Documents" of home
            set label index to 3
            set comment to "Smith Project Final"
        end tell
    end tell
```

Nested Tell Blocks: Multi-level

```
tell application "Finder"
    tell home
        tell folder "Documents"
            tell folder "Smith Project"
                tell every item
                    set label index to 3
                    set comment to "Smith Project Final"
                end tell
            end tell
        end tell
    end tell

    tell application "Finder"
        tell every item of folder "Smith Project" of folder "Documents" of home
            set label index to 3
            set comment to "Smith Project Final"
        end tell
    end tell
```

Nested Tell Blocks: Multi-level

```
tell application "Finder"
    tell home
        tell folder "Documents"
            tell folder "Smith Project"
                tell every item
                    set label index to 3
                    set comment to "Smith Project Final"
                end tell
            end tell
        end tell
    end tell

    tell application "Finder"
        tell every item of folder "Smith Project" of folder "Documents" of home
            set label index to 3
            set comment to "Smith Project Final"
        end tell
    end tell
```

Nested Tell Blocks: Multi-level

```
tell application "Finder"
    tell home
        tell folder "Documents"
            tell folder "Smith Project"
                tell every item
                    set label index to 3
                    set comment to "Smith Project Final"
                end tell
            end tell
        end tell
    end tell

    tell every item of folder "Smith Project" of folder "Documents" of home
        set label index to 3
        set comment to "Smith Project Final"
    end tell
end tell
```

Nested Tell Blocks: Other Apps

```
tell application "Finder"
    activate
    set this_folder to (choose folder with prompt "Pick the folder containing the JPEG images to
            process:")
    set these_files to every document file of this_folder whose file type is "JPEG" or name extension
            is "jpg"
    repeat with i from 1 to the count of these_files
        set this_file to (item i of these_files as alias)
        tell application "Image Events"
            launch
            set this_image to open this_file
            rotate this_image to angle 270
            save this_image with icon
            close this_image
        end tell
    end repeat
end tell
```

Nested Tell Blocks: Other Apps

```
tell application "Finder"
    activate
    set this_folder to (choose folder with prompt "Pick the folder containing the JPEG images to
            process:")
    set these_files to every document file of this_folder whose file type is "JPEG" or name extension
            is "jpg"
    repeat with i from 1 to the count of these_files
        set this_file to (item i of these_files as alias)
        tell application "Image Events"
            launch
            set this_image to open this_file
            rotate this_image to angle 270
            save this_image with icon
            close this_image
        end tell
    end repeat
end tell
```

Nested Tell Blocks: Other Apps

```
tell application "Finder"
    activate
    set this_folder to (choose folder with prompt "Pick the folder containing the JPEG images to
            process:")
    set these_files to every document file of this_folder whose file type is "JPEG" or name extension
            is "jpg"
    repeat with i from 1 to the count of these_files
        set this_file to (item i of these_files as alias)
        tell application "Image Events"
            launch
            set this_image to open this_file
            rotate this_image to angle 270
            save this_image with icon
            close this_image
        end tell
    end repeat
end tell
```

Nested Tell Blocks: Other Apps

```
tell application "Finder"
    activate
    set this_folder to (choose folder with prompt "Pick the folder containing the JPEG images to
            process:")
    set these_files to every document file of this_folder whose file type is "JPEG" or name extension
            is "jpg"
    repeat with i from 1 to the count of these_files
        set this_file to (item i of these_files as alias)
        tell application "Image Events"
            launch
            set this_image to open this_file
            rotate this_image to angle 270
            save this_image with icon
            close this_image
        end tell
    end repeat
end tell
```

Nested Tell Blocks: Other Apps

```
tell application "Finder"
    activate
    set this_folder to (choose folder with prompt "Pick the folder containing the JPEG images to
            process:")
    set these_files to every document file of this_folder whose file type is "JPEG" or name extension
            is "jpg"
    repeat with i from 1 to the count of these_files
        set this_file to (item i of these_files as alias)
        tell application "Image Events"
            launch
            set this_image to open this_file
            rotate this_image to angle 270
            save this_image with icon
            close this_image
        end tell
    end repeat
end tell
```

Nested Tell Blocks: Other Apps

```
tell application "Finder"
    activate
    set this_folder to (choose folder with prompt "Pick the folder containing the JPEG images to
            process:")
    set these_files to every document file of this_folder whose file type is "JPEG" or name extension
            is "jpg"
    repeat with i from 1 to the count of these_files
        set this_file to (item i of these_files as alias)
        tell application "Image Events"
            launch
            set this_image to open this_file
            rotate this_image to angle 270
            save this_image with icon
            close this_image
        end tell
    end repeat
end tell
```

Nested Tell Blocks: Other Apps

```
tell application "Finder"
    activate
    set this_folder to (choose folder with prompt "Pick the folder containing the JPEG images to
            process:")
    set these_files to every document file of this_folder whose file type is "JPEG" or name extension
            is "jpg"
    repeat with i from 1 to the count of these_files
        set this_file to (item i of these_files as alias)
        tell application "Image Events"
            launch
            set this_image to open this_file
            rotate this_image to angle 270
            save this_image with icon
            close this_image
        end tell
    end repeat
end tell
```

Nested Tell Blocks: Other Apps

```
tell application "Finder"
    activate
    set this_folder to (choose folder with prompt "Pick the folder containing the JPEG images to
            process:")
    set these_files to every document file of this_folder whose file type is "JPEG" or name extension
            is "jpg"
    repeat with i from 1 to the count of these_files
        set this_file to (item i of these_files as alias)
        tell application "Image Events"
            launch
            set this_image to open this_file
            rotate this_image to angle 270
            save this_image with icon
            close this_image
        end tell
    end repeat
end tell
```

Nested Tell Blocks: Other Apps

```
tell application "Finder"
    activate
    set this_folder to (choose folder with prompt "Pick the folder containing the JPEG images to
            process:")
    set these_files to every document file of this_folder whose file type is "JPEG" or name extension
            is "jpg"
    repeat with i from 1 to the count of these_files
        set this_file to (item i of these_files as alias)
        tell application "Image Events"
            launch
            set this_image to open this_file
            rotate this_image to angle 270
            save this_image with icon
            close this_image
        end tell
    end repeat
end tell
```

Nested Tell Blocks: Other Apps

```
tell application "Finder"
    activate
    set this_folder to (choose folder with prompt "Pick the folder containing the JPEG images to
            process:")
    set these_files to every document file of this_folder whose file type is "JPEG" or name extension
            is "jpg"
    repeat with i from 1 to the count of these_files
        set this_file to (item i of these_files as alias)
        tell application "Image Events"
            launch
            set this_image to open this_file
            rotate this_image to angle 270
            save this_image with icon
            close this_image
        end tell
    end repeat
end tell
```

Properties Record

```
tell application "Finder"
set the properties of the Finder preferences to {desktop shows hard disks:false,
          desktop shows removable media:false, desktop shows connected
          servers:false, folders open in new windows:false, new windows open in
          column view:false, all name extensions showing:true, folders spring
          open:true, delay before springing:0.668}
set warns before emptying of the trash to true
end tell
```

Properties Record

```
tell application "Finder"
set the properties of the Finder preferences to {desktop shows hard disks:false,
          desktop shows removable media:false, desktop shows connected
          servers:false, folders open in new windows:false, new windows open in
          column view:false, all name extensions showing:true, folders spring
          open:true, delay before springing:0.668}
set warns before emptying of the trash to true
end tell
```

Properties Record

```
tell application "Finder"
set the properties of the Finder preferences to {desktop shows hard disks:false,
          desktop shows removable media:false, desktop shows connected
          servers:false, folders open in new windows:false, new windows open in
          column view:false, all name extensions showing:true, folders spring
          open:true, delay before springing:0.668}
set warns before emptying of the trash to true
end tell
```

AppleScript Data Types

http://developer.apple.com/wwdc2005/

Data Types

Integer: 1, 5, 2345

Real: 1.0, 5.124, 2345.6789

String: "Bob drove the car."

Boolean: true, false

List: {"Sal", 123, alias "Macintosh HD:", "Fred", {"Sue", 34}}

Record: {key:value, key:value, key:value}

Operators

http://developer.apple.com/wwdc2005/

Math Operators: Addition

2 + 2 = 4

Math Operators: Multiply

2 * 2 = 4

Math Operators: Division

2 / 2 = 1

3 div 2 = 1

3 mod 2 = 1

Math Operators: Exponential

2 ^ 4 = 16

4 ^ 0.5 = 2

Concatenation: Text

"Once upon a time" & "in a village far away."

"Once upon a timein a village far away."

"Once upon a time" & space & "in a village far away."

"Once upon a time in a village far away."

Special terms: space, tab, return

Concatenation: Lists

{"Sal", "Sue", "Bob"} & {"Carl", "Wanda", "Zelda"}

{"Sal", "Sue", "Bob", "Carl", "Wanda", "Zelda"}

Coercions

Converting from one data format to another

http://developer.apple.com/wwdc2005/

Coercions

- Use the term “as” followed by the class to coerce to

```
"1" as integer
--> 1

1 as boolean
--> true

1 as Unicode text
--> "1"

"Fred" as list
--> {"Fred"}
```

Coercions

- Use text item delimiters when coercing lists to strings

```
{"Sal", "Sue", "Bob"} as Unicode text
--> "SalSueBob"

set AppleScript's text item delimiters to "-"
{"Sal", "Sue", "Bob"} as Unicode text
--> "Sal-Sue-Bob"
set AppleScript's text item delimiters to ""
```

Variables

AppleScript Data Containers

http://developer.apple.com/wwdc2005/

Variables

Any word or contiguous text that is not part of…

- the AppleScript language
- the terminologies of the installed scripting additions
- the current application terminology

Variables can contain any data type supported by AppleScript:
numeric, textual, lists, references, image data, booleans,
enumerations, records, etc.

Variables do not need to be declared or typed

All variables are local unless indicated otherwise

Variables

- Any word or contiguous text not part of the AppleScript language, the terminologies of the installed scripting additions, or the current application terminology

```
x
page12
this_project
theLastStrawberry
|Smith project file|
```

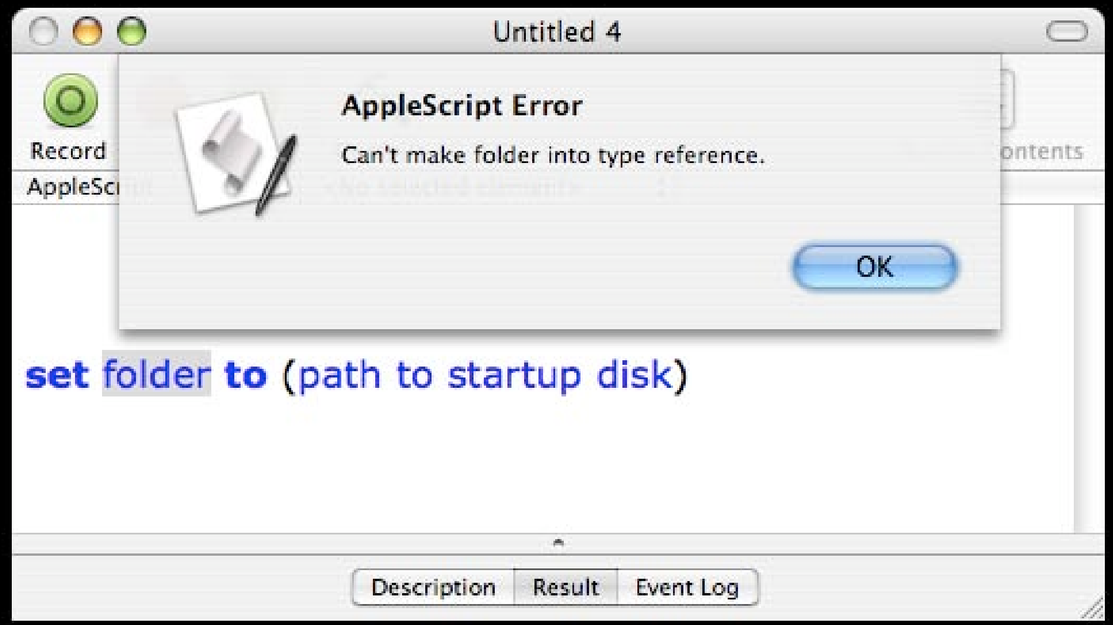
Variables

- Any word or contiguous text not part of the AppleScript language, the terminologies of the installed scripting additions, or the current application terminology

```
x
page12
this_project
theLastStrawberry
|Smith project file|
```

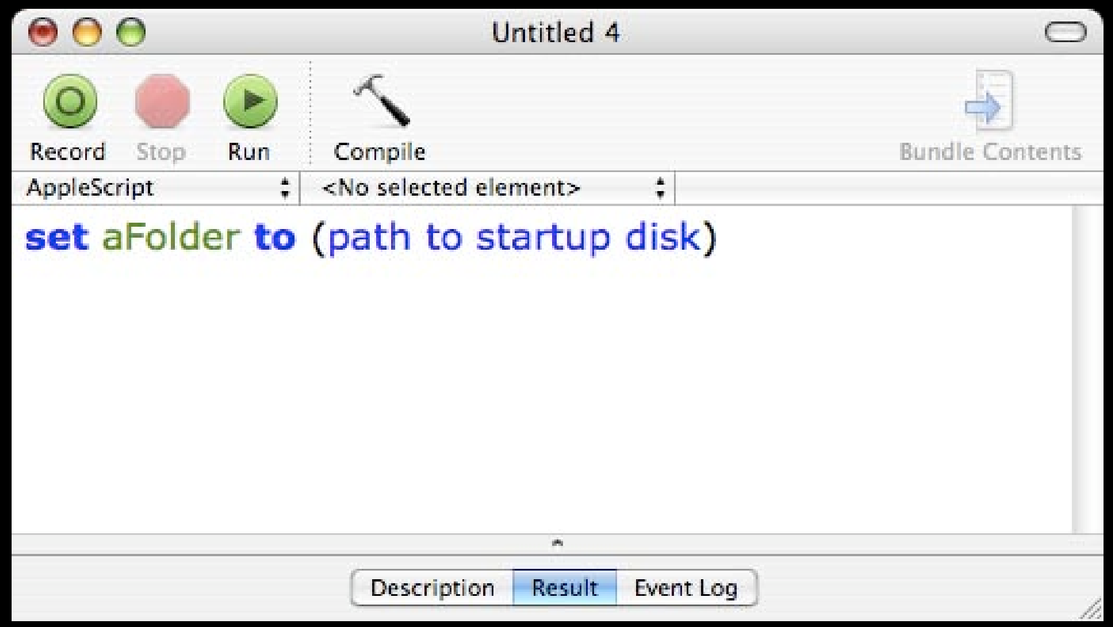
Variables

- Any word or contiguous text not part of the AppleScript language, **the terminologies of the installed scripting additions**, or the current application terminology



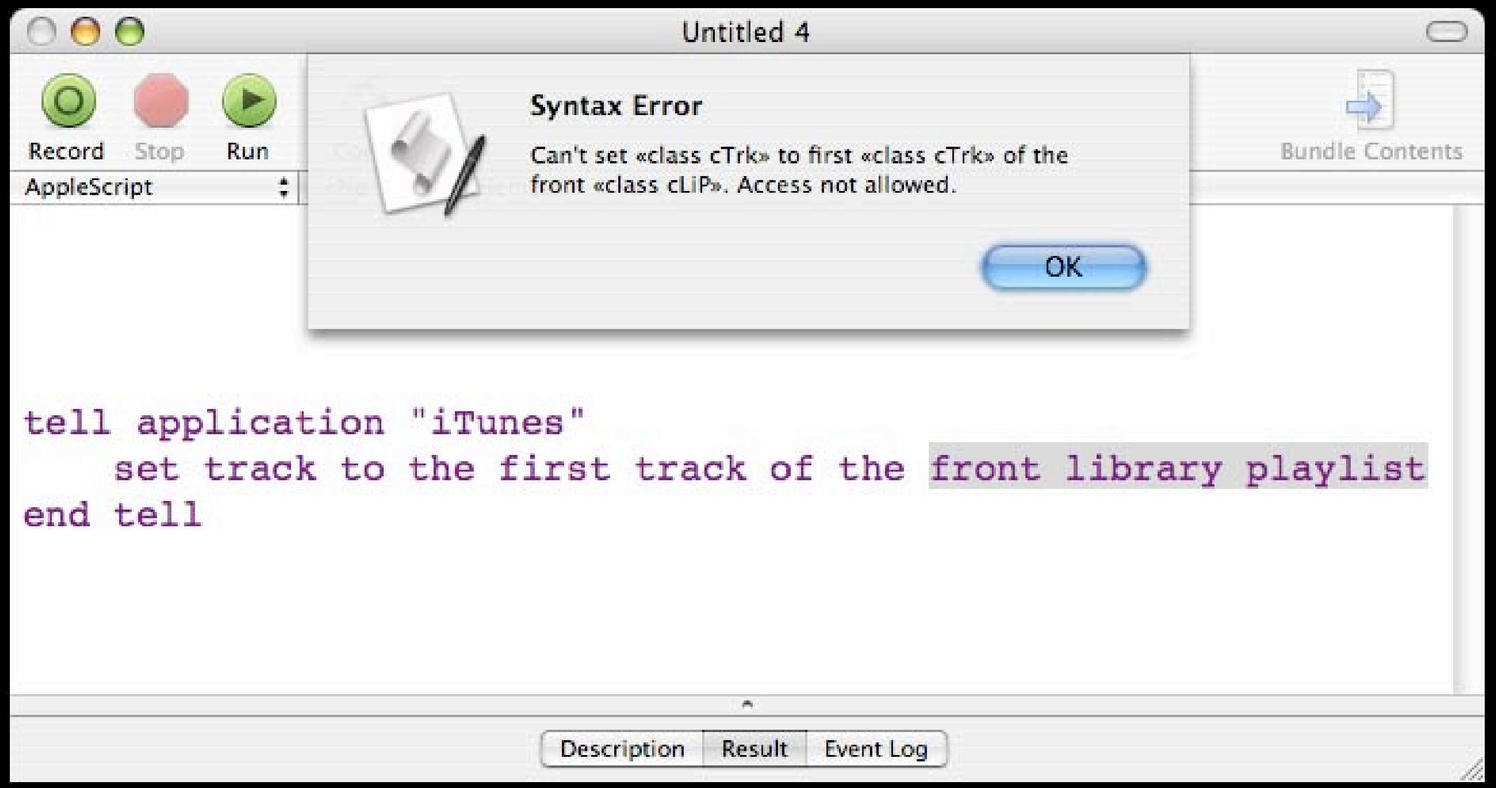
Variables

- Any word or contiguous text not part of the AppleScript language, **the terminologies of the installed scripting additions**, or the current application terminology



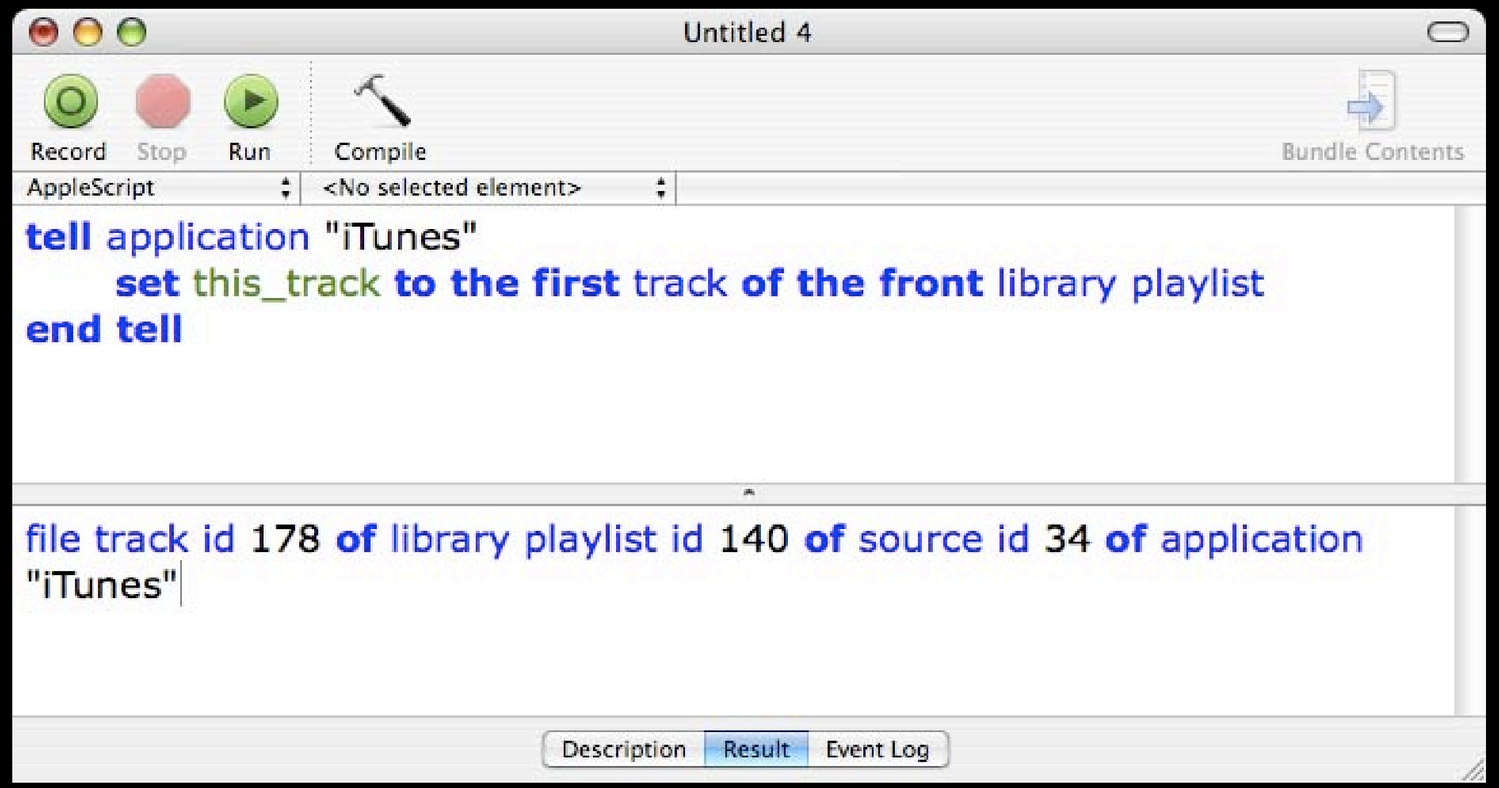
Variables

- Any word or contiguous text not part of the AppleScript language, the terminologies of the installed scripting additions, or the **current application terminology**



Variables

- Any word or contiguous text not part of the AppleScript language, the terminologies of the installed scripting additions, or the **current application terminology**



Variables

- Variables do not need to be declared or typed

```
set this_number to 45
set this_name to "Fred"
set this_disk to the path to the startup disk
set this_list to {1, "butter"}
```

Variables

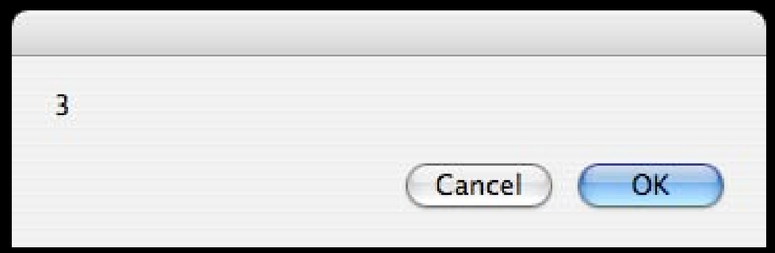
- Variables do not need to be declared or typed

```
copy {45, "Fred", path to startup disk, {1, "butter"}} to ¬
    {this_number, this_name, this_disk, this_list}

this_number is greater than 40
--> true

length of this_name
--> 4

this_disk as Unicode text
--> "Mac OS X:"

length of this_list
--> 2
```

Variables

- All variables are local unless declared to be global

```
set x to 3
setTheVariableTo("Wubba")
display dialog x

on setTheVariableTo(thisValue)
    set x to thisValue
end setTheVariable
```



Variables

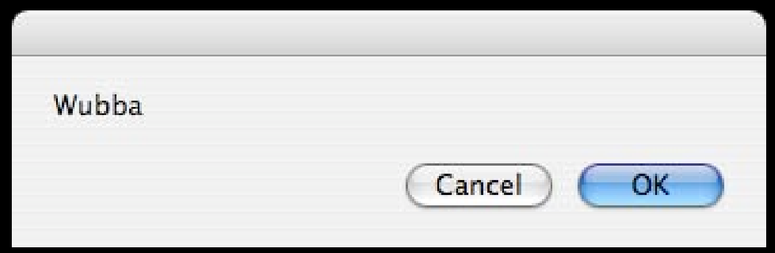
- All variables are local unless declared to be global

```
global x

set x to 3
setTheVariableTo("Wubba")
display dialog x

on setTheVariableTo(thisValue)
    set x to thisValue
end setTheVariable
```

Variables

- All variables are local unless declared to be global

```
global x

set x to 3
setTheVariableTo("Wubba")
display dialog x

on setTheVariableTo(thisValue)
    set x to thisValue
end setTheVariable
```



Item References

How to refer to scriptable items in AppleScript

http://developer.apple.com/wwdc2005/

Item References

Scriptable items are referenced by their position in their object hierarchy or where they are in their “Chain of Command.”

Item References

- Nested references
- Path references
- Alias references
- UNIX references (POSIX)
- File URLs

Nested References

- A nested reference described an item's position in it's object hierarchy from the bottom to the top of it's chain using the possessive "of" to indicate the containment of one item within another.
- Nested references are the default reference format returned by scriptable applications such as the Finder or System Events.
- Nested references are also used for scriptable items other than files

Nested Reference Examples

```
folder "Documents" of folder "username" of folder "Users" of startup disk

document file "cars.pdf" of folder "Documents" of folder "username" of
folder "Users" of startup disk

frame 13 of track "Video Overlay" of movie 2

text box 4 of page 2 of spread 13 of document 1

transition 3 of video track 2 of program 1 of project "TV"
```

Nested Reference Examples

```
folder "Documents" of folder "username" of folder "Users" of startup disk

document file "cars.pdf" of folder "Documents" of folder "username" of
folder "Users" of startup disk

frame 13 of track "Video Overlay" of movie 2

text box 4 of page 2 of spread 13 of document 1

transition 3 of video track 2 of program 1 of project "TV"
```

Path References

- Path References are used to pass a reference to a disk item to the Finder or System Events applications.
- They are text strings, read from left to right, that contain the names of each item in the object hierarchy, delimited by colon characters.
- Path references are preceded with the name of the class of the the target item such as: `disk`, `folder`, `document file`, or `item`
- Path references to folders and disks end with a colon

Path Reference Examples

```
tell application "Finder"
  disk "Macintosh HD:"
  folder "Macintosh HD:Users:username:Documents:"
  item "Macintosh HD:Users:username:Documents:"
  document file "Macintosh HD:Users:sal:Documents:ACC.pdf"
  item "Macintosh HD:Users:sal:Documents:ACC.pdf"
end tell
```

Path Reference Examples

```
tell application "Finder"
  disk "Macintosh HD:"
  folder "Macintosh HD:Users:username:Documents:"
  item "Macintosh HD:Users:username:Documents:"
  document file "Macintosh HD:Users:sal:Documents:ACC.pdf"
  item "Macintosh HD:Users:sal:Documents:ACC.pdf"
end tell
```

```
tell application "System Events"
  disk "Macintosh HD:"
  folder "Macintosh HD:Users:username:Documents:"
  item "Macintosh HD:Users:username:Documents:"
  file "Macintosh HD:Users:sal:Documents:ACC.pdf"
  item "Macintosh HD:Users:sal:Documents:ACC.pdf"
end tell
```

Path Reference Examples

```
tell application "Finder"
  disk "Macintosh HD:"
  folder "Macintosh HD:Users:username:Documents:"
  item "Macintosh HD:Users:username:Documents:"
  document file "Macintosh HD:Users:sal:Documents:ACC.pdf"
  item "Macintosh HD:Users:sal:Documents:ACC.pdf"
end tell
```

```
tell application "System Events"
  disk "Macintosh HD:"
  folder "Macintosh HD:Users:username:Documents:"
  item "Macintosh HD:Users:username:Documents:"
  file "Macintosh HD:Users:sal:Documents:ACC.pdf"
  item "Macintosh HD:Users:sal:Documents:ACC.pdf"
end tell
```

Alias References

- Recognized by all applications, alias references are used when passing file references between applications
- They are text strings, read from left to right, that contain the names of each item in the object hierarchy, delimited by colon characters.
- Alias references are preceded with the term: `alias`
- Alias references to folders and disks end with a colon
- Alias references point to existing items and are tracked by AppleScript
- Can be used within or outside of tell blocks

Alias Reference Examples

```
alias "Macintosh HD:"

alias "Macintosh HD:Users:sal:Documents:"

alias "Macintosh HD:Users:sal:Documents:ACC.pdf"
```

POSIX References

- Derived from the UNIX-based architecture of Mac OS X
- Generally not supported by most scriptable applications

POSIX Reference Examples

```
/Users/sal/Documents/

/Users/sal/Documents/ACC.pdf

/Volumes/Mac\ OS\ X/Backup/

'/Volumes/Mac OS X/Backup/'
```

POSIX References

- AppleScript provides a way for deriving POSIX references from other reference types: `POSIX path`
- AppleScript provides a way to coerce POSIX references to other reference types: `POSIX file`

Deriving POSIX Paths

```
POSIX path of alias "Macintosh HD:Users:sal:Documents:"
  --> returns: "/Users/sal/Documents/"
```

Deriving POSIX Paths

```
POSIX path of alias "Macintosh HD:Users:sal:Documents:"
  --> returns: "/Users/sal/Documents/"
```

Quoted Form

```
Quoted form of POSIX path of alias "Macintosh HD:Users:sal:Documents:"
  --> returns: "'/Users/sal/Documents/'"
```

Coercing POSIX Paths

```
"/Users/sal/Documents/" as POSIX file
  --> returns: file "Macintosh HD:Users:sal:Documents:"
```

Coercing POSIX Paths

```
"/Users/sal/Documents/" as POSIX file
  --> returns: file "Macintosh HD:Users:sal:Documents:"
```

Coercing POSIX Paths

```
"/Users/sal/Documents/" as POSIX file as alias
  --> returns: alias "Macintosh HD:Users:sal:Documents:"
```

File URLs

- File URLs describe an item in the same format used on the Internet
- Can be derived via the Finder and Systems Events apps

File URL Examples

```
tell application "Finder" to get the url of folder   "Documents" of home
--> returns: "file://localhost/Users/sal/Documents/"

"file://localhost/Users/sal/New%20Projects/ACC%20Scores.pdf"

"file://localhost/Volumes/Classic/Desktop%20Folder/"
```

File URL Examples

```
tell application "Finder" to get the url of folder "Documents" of home
--> returns: "file://localhost/Users/sal/Documents/"

"file://localhost/Users/sal/New%20Projects/ACC%20Scores.pdf"

"file://localhost/Volumes/Classic/Desktop%20Folder/"
```

Coercing Reference Types

- You can coerce the following file reference types to another file reference type:
  - nested reference to alias reference
  - path reference to alias reference
  - POSIX path to POSIX file to alias reference
  - alias reference to POSIX path

Coercing Reference Types

```
tell application "Finder"
  (folder "Applications" of the startup disk) as alias
  --> returns: alias "Macintosh HD:Applications:"

  (folder "Macintosh HD:Applications:") as alias
  --> returns: alias "Macintosh HD:Applications:"
end tell
```

```
POSIX path of alias "Macintosh HD:Applications:"
  --> returns: "/Applications/"

"/Applications/" as POSIX file as alias
  --> returns: alias "Macintosh HD:Applications:"
```

Finder Items as alias list

```
tell application "Finder"
    try
        every folder of the startup disk as alias list
    on error
        every folder of the startup disk as alias as list
    end try
    --> returns: {alias "Mac OS X:Applications:", alias "Mac OS
            X:Developer:", alias "Mac OS X:Library:", alias "Mac OS X:System:",
            alias "Mac OS X:Users:"}
end tell
```

Finder Items as alias list

```
tell application "Finder"
    try
        every folder of the startup disk as alias list
    on error
        every folder of the startup disk as alias as list
    end try
    --> returns: {alias "Mac OS X:Applications:", alias "Mac OS
            X:Developer:", alias "Mac OS X:Library:", alias "Mac OS X:System:",
            alias "Mac OS X:Users:"}
end tell
```

Finder Items as alias list

```
tell application "Finder"
    try
        every folder of the startup disk as alias list
    on error
        every folder of the startup disk as alias as list
    end try
    --> returns: {alias "Mac OS X:Applications:", alias "Mac OS
            X:Developer:", alias "Mac OS X:Library:", alias "Mac OS X:System:",
            alias "Mac OS X:Users:"}
end tell
```

Finder Items as alias list

```
tell application "Finder"
    try
        every folder of the startup disk as alias list
    on error
        every folder of the startup disk as alias as list
    end try
    --> returns: {alias "Mac OS X:Applications:", alias "Mac OS
          X:Developer:", alias "Mac OS X:Library:", alias "Mac OS X:System:",
          alias "Mac OS X:Users:"}
end tell
```

Demo

Sal Soghoian

These are confidential sessions—please refrain from filming or taking pictures.

Track-specific image to come

http://developer.apple.com/wwdc2003/urls.html

Finding Items

How to locate specific items using AppleScript

http://developer.apple.com/wwdc2005/

Finding Items

AppleScript can locate objects belonging to scriptable applications.

- tracks of an iTunes music library
- images of an iPhoto catalog
- text frames on a page in an InDesign document

Specific disk items can be located by querying the Finder or System Events applications.

Finding Items

Items can be located using two methods:

- Find by **Index**
- Find by **Property**

Finding Items by Index

- Every scriptable item on disk or belonging to an application occupies a specific place in the order of items in its parent container.
- The numeric value representing an item's place in a group of items is called its "**index**."

Finding Items by Index

```
tell application "Finder" to get item 1 of the startup disk
  --> returns: folder "Applications" of the startup disk

tell application "Finder" to get item 12 of home
  --> returns: folder "Downloads" of folder "sal" of folder "Users" of the
startup disk

tell application "Finder" to get item (count the items of home) of home
  --> returns: folder "USB Keys" of folder "sal" of folder "Users" of the
startup disk

{item 1, item 2, item 3, item 4, item 5, item 6}
```

Finding Items by Negative Index

- AppleScript supports the iteration of lists in both directions
- Items in a list have positive index values when the list is transversed from left to right (top to bottom), and negative index values when transversed from right to left (bottom to top)

Finding Items by Negative Index

```
L to R: {item 1, item 2, item 3, item 4, item 5}

R to L: {item -5, item -4, item -3, item -2, item -1}

tell application "Finder" to get item (count the items of home) of home
  --> returns: folder "USB Keys" of folder "sal" of folder "Users" of the
startup disk
```

```
tell application "Finder" to get item -1 of home
  --> returns: folder "USB Keys" of folder "sal" of folder "Users" of the
startup disk
```

Finding Items by Index Range

A range of items can be referenced by index values

```
tell application "Finder" to get items 2 thru 5 of the
startup disk
  --> returns: {folder "Applications" of startup disk of
application "Finder", alias file "Desktop (Mac OS 9)" of
startup disk of application "Finder", folder "Developer" of
startup disk of application "Finder", folder "Disk Images" of
startup disk of application "Finder"}

tell application "Finder" to get items 1 thru -1 of the
startup disk

tell application "Finder" to get every item of the startup
disk
```

Finding Items by Descriptive Index

Index values can be described in “conversational” style, except when used in ranges

```
{first, second, third, fourth, fifth, sixth, seventh, eighth,
ninth, tenth}

tell application "Finder" to get the first item of home
```

```
1st, 23rd, 10th, 32nd, 4068th

tell application "Finder" to get the 22nd item of home
```

Finding Items by Relative Position

Items can be referenced in terms of their position relative to other items in their container or list

```
front, first, back, last, middle

the last item of the startup disk
the middle track of the front library

after, before

the item before the last item
the item after the middle item
```

Finding Random Items

Items can be located randomly

```
some

some item of the startup disk
```

Finding Items by Property

- Every scriptable item on disk or belonging to an application has properties whose values can be used to locate the item.
  - name
  - kind (file type)
  - size
  - creation | modification date

Finding Items by Property

Items can be located by comparing the value of their properties to a desired set of values

```
the first document file whose name contains "Smith Project"

the first folder whose size is greater than 500000

every document file whose file type is "JPEG"

every document file whose name extension is "pdf"
```

Finding Items by Property

- A query is constructed by combining:
  - Positional Indicator
  - Target Object Type
  - Target Container
  - Object Specifier
  - Target Property
  - Comparison Operator
  - Target Property Value

Finding Items by Property

- A query is constructed by combining:
  - Positional Indicator
  - Target Object Type
  - Target Container
  - Object Specifier
  - Target Property
  - Comparison Operator
  - Target Property Value

```
the first document file of home whose name contains "Smith"
```

Finding Items by Property

- A query is constructed by combining:
  - **Positional Indicator**
  - Target Object Type
  - Target Container
  - Object Specifier
  - Target Property
  - Comparison Operator
  - Target Property Value

```
the first document file of home whose name contains "Smith"
```

Finding Items by Property

- A query is constructed by combining:
  - Positional Indicator
  - **Target Object Type**
  - **Target Container**
  - Object Specifier
  - Target Property
  - Comparison Operator
  - Target Property Value

```
the first document file of home whose name contains "Smith"
```

Finding Items by Property

- A query is constructed by combining:
  - Positional Indicator
  - Target Object Type
  - **Target Container**
  - Object Specifier
  - Target Property
  - Comparison Operator
  - Target Property Value

```
the first document file of home whose name contains "Smith"
```

Finding Items by Property

- A query is constructed by combining:
  - Positional Indicator
  - Target Object Type
  - Target Container
  - **Object Specifier**
  - Target Property
  - Comparison Operator
  - Target Property Value

```
the first document file of home whose name contains "Smith"
```

Finding Items by Property

- A query is constructed by combining:
  - Positional Indicator
  - Target Object Type
  - Target Container
  - Object Specifier
  - **Target Property**
  - Comparison Operator
  - Target Property Value

```
the first document file of home whose name contains "Smith"
```

Finding Items by Property

- A query is constructed by combining:
  - Positional Indicator
  - Target Object Type
  - Target Container
  - Object Specifier
  - Target Property
  - **Comparison Operator**
  - Target Property Value

```
the first document file of home whose name contains "Smith"
```

Finding Items by Property

- A query is constructed by combining:
  - Positional Indicator
  - Target Object Type
  - Target Container
  - Object Specifier
  - Target Property
  - Comparison Operator
  - **Target Property Value**

```
the first document file of home whose name contains "Smith"
```

Finding Items by Property

- A query is constructed by combining:
  - Positional Indicator
  - Target Object Type
  - Target Container
  - Object Specifier
  - Target Property
  - Comparison Operator
  - Property Value

```
the first document file of home whose name contains "Smith"
```

Finding Items by Property

Changing the Positional Indicator:

```
the first document file whose name contains "Smith Project"

the last document file whose name contains "Smith Project"

every document file whose name contains "Smith Project"
```

Finding Items by Property

Changing the Comparison Operator:

```
every document file whose name contains "Smith Project"

every document file whose name does not contain "Smith
Project"

every document file whose name begins with "Smith Project"

every document file whose name ends with "Smith Project"
```

Finding Items by Property

Changing the Property Value:

```
every document file whose name contains "Smith Project"

every document file whose name contains "ACC Stats"
```

Finding Items by Property

Searching sub-folders:

```
every document file of the entire contents of folder
"Documents" of home whose name contains "Smith Project"
```

Compound Property Queries

You can further define your search by adding additional property conditions using “and”:

```
tell application "Finder" to get the first item of the
startup disk whose name begins with "S" and name ends with
"folder"
  --> returns: folder "System folder" of startup disk
```

```
tell application "Finder" to get every document file whose
name extension is in {"jpg", "jpeg"} and name contains "Smith
Project"
```

Compound Property Queries

You can further define your search by adding optional property conditions using “or”:

```
tell application "Finder" to get every document file of
folder "Pictures" of home whose file type is "JPEG" or name
extension is in {"jpg", "jpeg"}
```

Compound Property Queries

You can further define your search by adding optional property conditions using “or” and “and”:

```
tell application "Finder" to get every document file of
folder "Pictures" of home whose (file type is "JPEG" or name
extension is in {"jpg", "jpeg"}) and name contains "Smith
Project"
```

Compound Property Queries

You can further define your search by adding optional property conditions using “or” and “and”:

```
tell application "Finder" to get every document file of
folder "Pictures" of home whose (file type is "JPEG" or name
extension is in {"jpg", "jpeg"}) and name contains "Smith
Project"
```

Demo

Sal Soghoian

These are confidential sessions—please refrain from filming or taking pictures.

Track-specific image to come

http://developer.apple.com/wwdc2003/urls.html

Special Folders

Mac OS X Special Folder Designations

http://developer.apple.com/wwdc2005/

Special Folders

Special folders provide a generic means of locating specific directories in the standard Mac OS X file system.

Special Finder Folders

The Finder application has four special folders:

```
startup disk, home, desktop, trash

tell application "Finder" to delete every item of the desktop
whose name contains "Smith Project"

tell application "Finder" to duplicate every folder of the
folder "Projects" of home whose name begins with "Smith
Project" to the folder named "Backup"

tell application "Finder" to empty the trash
```

The "path to" Command

- The `path to` command of the Standard Additions scripting addition can be used to locate dozens of special folders in any of four domains:
  - user domain
  - local domain
  - System domain
  - network domain

The "path to" Command

- The `path to` command of the Standard Additions scripting addition can be used to locate dozens of special folders in any of four domains:
  - **user domain**
  - local domain
  - System domain
  - network domain

```
path to library folder from user domain
--> returns: alias "Macintosh HD:Users:sal:Library:"
```

The "path to" Command

- The `path to` command of the Standard Additions scripting addition can be used to locate dozens of special folders in any of four domains:
  - user domain
  - **local domain**
  - System domain
  - network domain

```
path to library folder from local domain
--> returns: alias "Macintosh HD:Library:"
```

The "path to" Command

- The `path to` command of the Standard Additions scripting addition can be used to locate dozens of special folders in any of four domains:
  - user domain
  - local domain
  - **System domain**
  - network domain

```
path to library folder from System domain
--> returns: alias "Macintosh HD:System:Library:"
```

The "path to" Command

- The `path to` command of the Standard Additions scripting addition can be used to locate dozens of special folders in any of four domains:
  - user domain
  - local domain
  - System domain
  - network domain

```
path to library folder from network domain
--> returns: alias "Macintosh HD:Network:Library:"
```

The ”path to” Command

- Common folder codes and enumerations:

Documents folder	docs	documents folder	user*
Applications folder	apps	applications folder	user*, local, network
Movies folder	mdoc	movies folder	user*
Pictures folder	pdoc	pictures folder	user*
Music folder	µdoc	music folder	user*
Preferences folder	pref	preferences folder	user*, local
Public folder	pubb	public folder	user*
Scripts folder	scrƒ	scripts folder	users*, local
Sites folder	site	sites folder	user*
Users folder	usrs	users folder	local*
Utilities folder	utiƒ	utilities folder	user*, local
Fonts folder	font	fonts folder	user*, local

The "path to" Command

- Common folder codes and enumerations:

Mac OS X v. 10.3

Documents folder	docs	documents folder	user*
Applications folder	apps	applications folder	user*, local, network
Movies folder	mdoc	movies folder	user*
Pictures folder	pdoc	pictures folder	user*
Music folder	µdoc	music folder	user*
Preferences folder	pref	preferences folder	user*, local
Public folder	pubb	public folder	user*
Scripts folder	scr*ƒ*	scripts folder	users*, local
Sites folder	site	sites folder	user*
Users folder	usrs	users folder	local*
Utilities folder	uti*ƒ*	utilities folder	user*, local
Fonts folder	font	fonts folder	user*, local

The ”path to” Command

application support • applications folder • desktop • desktop pictures folder • documents folder • favorites folder • Folder Action scripts • fonts • help • home folder • internet plugins • keychain folder • library folder • modem scripts • movies folder • music folder • pictures folder • preferences • printer descriptions • public folder • scripting additions • scripts folder • shared documents • shared libraries • sites folder • startup disk • startup items • system folder • system preferences • temporary items • trash • users folder • utilities folder • workflows folder • voices • apple menu • control panels • control strip modules • extensions • launcher items folder • printer drivers • printmonitor • shutdown folder • speakable items • stationery

The ”path to” Command

To create a reference to an item within a special folder:

```
set the parent_folder to (path to documents folder) as
Unicode text
  --> returns: "Macintosh HD:Users:sal:Documents"

set the item_path to the parent_folder & "New Projects:ACC
Stats.pdf"
  --> returns: "Macintosh HD:Users:sal:Documents:New
Projects:ACC Stats.pdf"

set this_item to the item_path as alias
  --> returns: alias "Macintosh HD:Users:sal:Documents:New
Projects:ACC Stats.pdf"
```

The ”path to” Command

To create a reference to an item within a special folder:

```
set this_item to ((path to documents folder as Unicode text)
    & "New Projects:ACC Stats.pdf") as alias

  --> returns: alias "Macintosh HD:Users:sal:Documents:New
Projects:ACC Stats.pdf"
```

The ”path to” Command

To create a reference to an item within a special folder:

```
set this_item to ((path to documents folder as Unicode text)
    & "New Projects:ACC Stats.pdf") as alias

  --> returns: alias "Macintosh HD:Users:sal:Documents:New
Projects:ACC Stats.pdf"
```

The ”path to” Command

The System Events application supports direct use of special folders designations:

```
tell application "System Events"
  set this_item to (item "ACC Stats.pdf" of folder "New
Projects" of the documents folder) as alias
  --> returns: alias "Macintosh HD:Users:sal:Documents:New
Projects:ACC Stats.pdf"
end tell
```

Demo

Sal Soghoian

These are confidential sessions—please refrain from filming or taking pictures.

Track-specific image to come

http://developer.apple.com/wwdc2003/urls.html

Info Tools

http://developer.apple.com/wwdc2005/

Additional Tools

- Locating and identifying items can be done with these additional tools:
  - list disks command
  - list folder command
  - info for command
  - system info

Additional Tools: “list disks”

The “list disks” command returns the name of the mounted volumes:

```
list disks
  --> returns: {"Macintosh HD", "Mac OS X", "Classic",
"Client Backup"}
```

Additional Tools: “list folder”

The “list folder” command returns the name of every item in a specified folder:

```
list folder (path to startup disk) without invisibles
  --> returns: {"Applications", "Developer", "System",
"Users"}

list folder (path to startup disk) with invisibles
  --> returns: {".vol", ".DSStore", "Applications",
"Developer", "System", "Users"}
```

Additional Tools: “list folder”

The “list folder” command returns the name of every item in a specified folder:

```
list folder (path to startup disk) without invisibles
  --> returns: {"Applications", "Developer", "System",
"Users"}

list folder (path to startup disk) with invisibles
  --> returns: {".vol", ".DSStore", "Applications",
"Developer", "System", "Users"}
```

Additional Tools: “info for”

The “info for” command returns a record of key properties of a specified item:

```
info for ((path to desktop folder) as Unicode text &
"AppleScript & Java.pdf") as alias
  --> returns: {name:"AppleScript & Java.pdf", creation
date:date "Wednesday, February 26, 2003 6:42:52 PM",
modification date:date "Wednesday, February 26, 2003 6:42:54
PM", icon position:{129, 448}, visible:true, size:3.43188E+5,
folder:false, alias:false, name extension:"pdf", extension
hidden:false, package folder:false, file type:"PDF ", file
creator:"CARO", displayed name:"AppleScript & Java.pdf",
default application:alias "Macintosh HD:Applications:Adobe
Acrobat 5.0:Acrobat 5.0", locked:false, busy status:false,
short version:"", long version:"", type
identifier:"com.adobe.pdf"}
```

Additional Tools: “info for”

The “info for” command returns a record of key properties of a specified item:

```
info for ((path to desktop folder) as Unicode text &
"AppleScript & Java.pdf") as alias
  --> returns: {name:"AppleScript & Java.pdf", creation
date:date "Wednesday, February 26, 2003 6:42:52 PM",
modification date:date "Wednesday, February 26, 2003 6:42:54
PM", icon position:{129, 448}, visible:true, size:3.43188E+5,
folder:false, alias:false, name extension:"pdf", extension
hidden:false, package folder:false, file type:"PDF ", file
creator:"CARO", displayed name:"AppleScript & Java.pdf",
default application:alias "Macintosh HD:Applications:Adobe
Acrobat 5.0:Acrobat 5.0", locked:false, busy status:false,
short version:"", long version:"", type
identifier:"com.adobe.pdf"}
```

Additional Tools: “info for”

The “info for” command returns a record of key properties of a specified item:

```
info for ((path to desktop folder) as Unicode text &
"AppleScript & Java.pdf") as alias
  --> returns: {name:"AppleScript & Java.pdf", creation
date:date "Wednesday, February 26, 2003 6:42:52 PM",
modification date:date "Wednesday, February 26, 2003 6:42:54
PM", icon position:{129, 448}, visible:true, size:3.43188E+5,
folder:false, alias:false, name extension:"pdf", extension
hidden:false, package folder:false, file type:"PDF ", file
creator:"CARO", displayed name:"AppleScript & Java.pdf",
default application:alias "Macintosh HD:Applications:Adobe
Acrobat 5.0:Acrobat 5.0", locked:false, busy status:false,
short version:"", long version:"", type
identifier:"com.adobe.pdf"}
```

Additional Tools: “system info”

The “system info” command returns a record of important properties of the computer and system:

```
system info
  --> returns: {AppleScript version:"1.10", AppleScript
Studio version:"1.4", system version:"10.4.1", short user
name:"sal", long user name:"Sal Soghoian", user ID:501, user
locale:"en_US", home directory:alias "Mac OS X:Users:sal:",
boot volume:"Mac OS X", computer name:"Sal-PB", host
name:"Sal-PB.local", IPv4 address:"127.0.0.1", primary
Ethernet address:"00:0a:95:a4:dc:92", CPU type:"PowerPC
7450", CPU speed:1249, physical memory:1024}
```

Additional Tools: “system info”

The “system info” command returns a record of important properties of the computer and system:

```
system info
  --> returns: {AppleScript version:"1.10", AppleScript
Studio version:"1.4", system version:"10.4.1", short user
name:"sal", long user name:"Sal Soghoian", user ID:501, user
locale:"en_US", home directory:alias "Mac OS X:Users:sal:",
boot volume:"Mac OS X", computer name:"Sal-PB", host
name:"Sal-PB.local", IPv4 address:"127.0.0.1", primary
Ethernet address:"00:0a:95:a4:dc:92", CPU type:"PowerPC
7450", CPU speed:1249, physical memory:1024}
```

“Clean and Wax”

Actions with found items

http://developer.apple.com/wwdc2005/

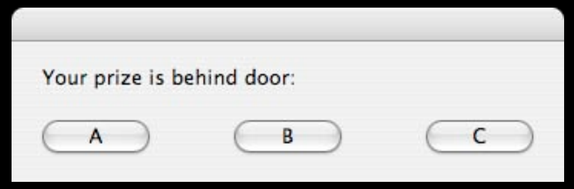
Conditionals

Single and Multiple Conditionals

http://developer.apple.com/wwdc2005/

Conditional Statement

```
display dialog "Your prize is behind door:" buttons {"A", "B", "C"}
```




```
--> {button returned:"A"}
```

Conditional Statement

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
if the button returned of the result is "A" then say "No prize here!"
```

Conditional Statement

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
if the button returned of the result is "A" then say "No prize here!"
```

Conditional Statement

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
if the button returned of the result is "A" then say "No prize here!"
```

Conditional Statement

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
if the button returned of the result is "A" then say "No prize here!"
```

Conditional Statement

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
if the button returned of the result is "A" then say "No prize here!"
```

Conditional Statement

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
if the button returned of the result is "A" then say "No prize here!"
```

Conditional Statement

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
if the button returned of the result is "A" then say "No prize here!"
```

Conditional Statement

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
if the button returned of the result is not "B" then say "No prize here!"
```

Conditional Statement

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
if the button returned of the result is not "B" then say "No prize here!"
```

Multiple Conditions

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
if the button returned of the result is not "B" then
   say "No prize here!"
else
   say "You won a car!"
end if
```

Multiple Conditions

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
if the button returned of the result is not "B" then
   say "No prize here!"
else
   say "You won a car!"
end if
```

Multiple Conditions

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
if the button returned of the result is not "B" then
   say "No prize here!"
else
   say "You won a car!"
end if
```

Multiple Conditions

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
if the button returned of the result is not "B" then
   say "No prize here!"
else
   say "You won a car!"
end if
```

Multiple Conditions

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
if the button returned of the result is not "B" then
   say "No prize here!"
else
   say "You won a car!"
end if
```

Multiple Conditions

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
if the button returned of the result is not "B" then
   say "No prize here!"
else
   say "You won a car!"
end if
```

Multiple Conditions

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
set the button_pressed to the button returned of the result
if the button_pressed is "A" then
   say "You won a pizza!"
else if the button_pressed is "B" then
   say "You won an iPod!"
else
   say "You won a car!"
end if
```

Multiple Conditions

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
set the button_pressed to the button returned of the result
if the button_pressed is "A" then
   say "You won a pizza!"
else if the button_pressed is "B" then
   say "You won an iPod!"
else
   say "You won a car!"
end if
```

Multiple Conditions

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
set the button_pressed to the button returned of the result
if the button_pressed is "A" then
   say "You won a pizza!"
else if the button_pressed is "B" then
   say "You won an iPod!"
else
   say "You won a car!"
end if
```

Multiple Conditions

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
set the button_pressed to the button returned of the result
if the button_pressed is "A" then
   say "You won a pizza!"
else if the button_pressed is "B" then
   say "You won an iPod!"
else
   say "You won a car!"
end if
```

Multiple Conditions

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
set the button_pressed to the button returned of the result
if the button_pressed is "A" then
   say "You won a pizza!"
else if the button_pressed is "B" then
   say "You won an iPod!"
else
   say "You won a car!"
end if
```

Multiple Conditions

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
set the button_pressed to the button returned of the result
if the button_pressed is "A" then
   say "You won a pizza!"
else if the button_pressed is "B" then
   say "You won an iPod!"
else
   say "You won a car!"
end if
```

Multiple Conditions

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
set the button_pressed to the button returned of the result
if the button_pressed is "A" then
   say "You won a pizza!"
else if the button_pressed is "B" then
   say "You won an iPod!"
else
   say "You won a car!"
end if
```

Multiple Conditions

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
set the button_pressed to the button returned of the result
if the button_pressed is "A" then
   say "You won a pizza!"
else if the button_pressed is "B" then
   say "You won an iPod!"
else
   say "You won a car!"
end if
```

Multiple Conditions

```
display dialog "The prize is behind door:" buttons {"A", "B", "C"}
set the button_pressed to the button returned of the result
if the button_pressed is "A" then
   say "You won a pizza!"
else if the button_pressed is "B" then
   say "You won an iPod!"
else
   say "You won a car!"
end if
```

Repeat Loops

Iterating lists and repeating actions

http://developer.apple.com/wwdc2005/

Repeat <n> times

```
repeat 4 times
   beep
   delay 1
end repeat
```

Repeat <n> times

```
repeat 4 times
   beep
   delay 1
end repeat
```

Repeat <n> times

```
repeat 4 times
   beep
   delay 1
end repeat
```

Repeat <while…>

```
set this_number to ""
repeat while this_number mod 2 is 0
   set this_number to random number from 1 to 10
end repeat
```

Repeat <while…>

```
set this_number to ""
repeat while this_number mod 2 is 0
   set this_number to random number from 1 to 10
end repeat
```

Repeat <while…>

```
set this_number to ""
repeat while this_number mod 2 is 0
   set this_number to random number from 1 to 10
end repeat
```

Repeat <until…>

```
set this_number to ""
repeat until this_number mod 2 is 0
   set this_number to random number from 1 to 10
end repeat
```

Repeat <until…>

```
set this_number to ""
repeat until this_number mod 2 is 0
   set this_number to random number from 1 to 10
end repeat
```

Repeat <until…>

```
set this_number to ""
repeat until this_number mod 2 is 0
   set this_number to random number from 1 to 10
end repeat
```

Exit a loop

```
set this_number to ""
repeat
   set this_number to random number from 1 to 10
   if this_number is greater than 8 then exit repeat
end repeat
```

Exit a loop

```
set this_number to ""
repeat
   set this_number to random number from 1 to 10
   if this_number is greater than 8 then exit repeat
end repeat
```

Exit a loop

```
set this_number to ""
repeat
   set this_number to random number from 1 to 10
   if this_number is greater than 8 then exit repeat
end repeat
```

Exit a loop

```
set this_number to ""
repeat
   set this_number to random number from 1 to 10
   if this_number is greater than 8 then exit repeat
end repeat
```

Repeat with <variable> in list

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

set the S_names to {}
repeat with this_friend in the friends_list
   if the first character of this_friend is "S" then
      set the end of the S_names to this_friend
   end if
end repeat
```

Repeat with <variable> in list

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

set the S_names to {}
repeat with this_friend in the friends_list
   if the first character of this_friend is "S" then
      set the end of the S_names to this_friend
   end if
end repeat
```

Repeat with <variable> in list

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

set the S_names to {}
repeat with this_friend in the friends_list
   if the first character of this_friend is "S" then
      set the end of the S_names to this_friend
   end if
end repeat
```

Repeat with <variable> in list

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

set the S_names to {}
repeat with this_friend in the friends_list
   if the first character of this_friend is "S" then
      set the end of the S_names to this_friend
   end if
end repeat
```

Repeat with <variable> in list

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

set the S_names to {}
repeat with this_friend in the friends_list
   if the first character of this_friend is "S" then
      set the end of the S_names to this_friend
   end if
end repeat
```

Repeat with <variable> in list

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

set the S_names to {}
repeat with this_friend in the friends_list
   if the first character of this_friend is "S" then
      set the end of the S_names to this_friend
   end if
end repeat
```

Repeat with <variable> in list

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

set the S_names to {}
repeat with this_friend in the friends_list
   if the first character of this_friend is "S" then
      set the end of the S_names to this_friend
   end if
end repeat
```

Repeat with <variable> in list

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

set the S_names to {}
repeat with this_friend in the friends_list
   if the first character of this_friend is "S" then
      set the end of the S_names to this_friend
   end if
end repeat
get the S_names
--> {item 1 of {"Sal", "Sue", "Carl", "Scarlett"}, item 2 of {"Sal", "Sue",
   "Carl", "Scarlett"}, item 4 of {"Sal", "Sue", "Carl", "Scarlett"}}
```

Repeat with <variable> in list

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

set the S_names to {}
repeat with this_friend in the friends_list
   if the first character of this_friend is "S" then
      set the end of the S_names to this_friend
   end if
end repeat
get the S_names
--> {item 1 of {"Sal", "Sue", "Carl", "Scarlett"}, item 2 of {"Sal", "Sue",
   "Carl", "Scarlett"}, item 4 of {"Sal", "Sue", "Carl", "Scarlett"}}

set the S_names to {}
repeat with this_friend in the friends_list
   if the first character of this_friend is "S" then
      set the end of the S_names to the contents of this_friend
   end if
end repeat
```

Repeat with <variable> in list

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

set the S_names to {}
repeat with this_friend in the friends_list
   if the first character of this_friend is "S" then
      set the end of the S_names to this_friend
   end if
end repeat
get the S_names
--> {item 1 of {"Sal", "Sue", "Carl", "Scarlett"}, item 2 of {"Sal", "Sue",
   "Carl", "Scarlett"}, item 4 of {"Sal", "Sue", "Carl", "Scarlett"}}

set the S_names to {}
repeat with this_friend in the friends_list
   if the first character of this_friend is "S" then
      set the end of the S_names to the contents of this_friend
   end if
end repeat
get the S_names
--> {"Sal", "Sue", "Scarlett"}
```

Repeat with <loop-counter> from <x> to <y>

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

set the S_names to {}
repeat with i from 1 to the count of the friends_list
   set this_friend to item i of the friends_list
   if the first character of this_friend is "S" then
      set the end of the S_names to this_friend
   end if
end repeat
```

Repeat with <loop-counter> from <x> to <y>

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

set the S_names to {}
repeat with i from 1 to the count of the friends_list
   set this_friend to item i of the friends_list
   if the first character of this_friend is "S" then
      set the end of the S_names to this_friend
   end if
end repeat
```

Repeat with <loop-counter> from <x> to <y>

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

set the S_names to {}
repeat with i from 1 to the count of the friends_list
   set this_friend to item i of the friends_list
   if the first character of this_friend is "S" then
      set the end of the S_names to this_friend
   end if
end repeat
```

Repeat with <loop-counter> from <x> to <y>

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

set the S_names to {}
repeat with i from 1 to the count of the friends_list
   set this_friend to item i of the friends_list
   if the first character of this_friend is "S" then
      set the end of the S_names to this_friend
   end if
end repeat
```

Repeat with <loop-counter> from <x> to <y>

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

set the S_names to {}
repeat with i from 1 to the count of the friends_list
   set this_friend to item i of the friends_list
   if the first character of this_friend is "S" then
      set the end of the S_names to this_friend
   end if
end repeat
```

Repeat with <loop-counter> from <x> to <y>

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

set the S_names to {}
repeat with i from 1 to the count of the friends_list
   set this_friend to item i of the friends_list
   if the first character of this_friend is "S" then
      set the end of the S_names to this_friend
   end if
end repeat
```

Repeat with <loop-counter> from <x> to <y>

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

set the S_names to {}
repeat with i from 1 to the count of the friends_list
   set this_friend to item i of the friends_list
   if the first character of this_friend is "S" then
      set the end of the S_names to this_friend
   end if
end repeat
```

Repeat with <loop-counter> from <x> to <y>

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

set the S_names to {}
repeat with i from 1 to the count of the friends_list
   set this_friend to item i of the friends_list
   if the first character of this_friend is "S" then
      set the end of the S_names to this_friend
   end if
end repeat
```

Repeat with <loop-counter> from <x> to <y>

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

set the S_names to {}
repeat with i from 1 to the count of the friends_list
   set this_friend to item i of the friends_list
   if the first character of this_friend is "S" then
      set the end of the S_names to this_friend
   end if
end repeat
```

Repeat with <loop-counter> from <x> to <y>

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

set the S_names to {}
repeat with i from 1 to the count of the friends_list
   set this_friend to item i of the friends_list
   if the first character of this_friend is "S" then
      set the end of the S_names to this_friend
   end if
end repeat
```

Repeat with <loop-counter> from <x> to <y>

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

set the S_names to {}
repeat with i from 1 to the count of the friends_list
   set this_friend to item i of the friends_list
   if the first character of this_friend is "S" then
      set the end of the S_names to this_friend
   end if
end repeat
get the S_names
--> {"Sal", "Sue", "Scarlett"}
```

Repeat with <loop-counter> from <x> to <y>

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

repeat with i from 1 to the count of the friends_list
   display dialog (item i of the friends_list)
end repeat
```

Repeat with <loop-counter> from <x> to <y>

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

repeat with i from 1 to the count of the friends_list
   display dialog (item i of the friends_list)
end repeat
```

Repeat with <loop-counter> from <x> to <y>

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

repeat with i from 1 to the count of the friends_list
   display dialog (item i of the friends_list)
end repeat
```

Repeat with <loop-counter> from <x> to <y>

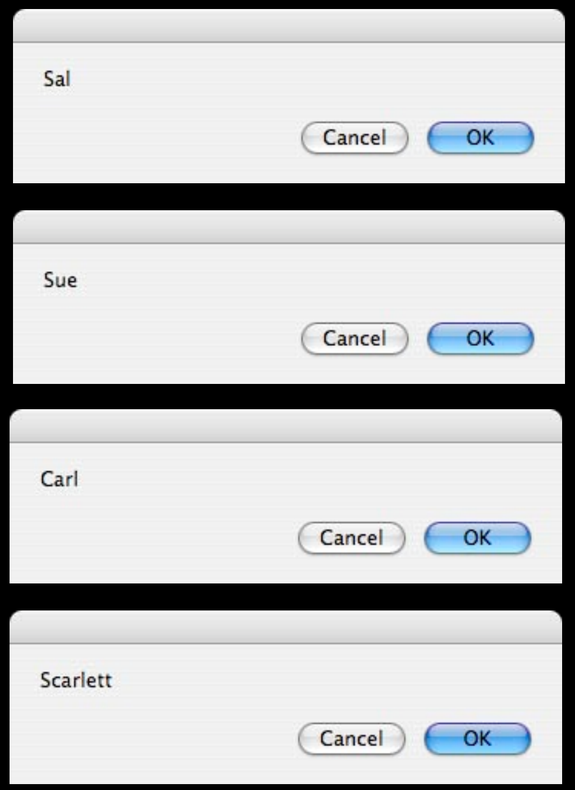
```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

repeat with i from 1 to the count of the friends_list
   display dialog (item i of the friends_list)
end repeat
```

Repeat with <loop-counter> from <x> to <y>

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

repeat with i from 1 to the count of the friends_list
   display dialog (item i of the friends_list)
end repeat
```

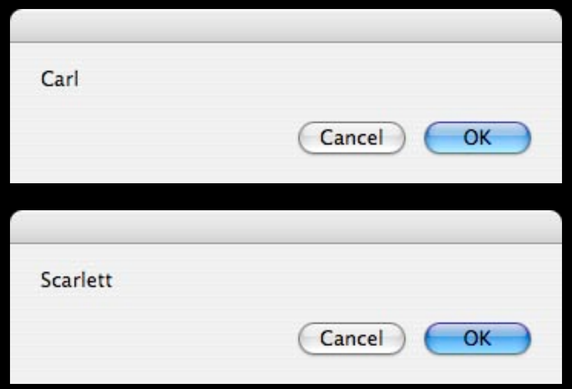
Repeat with <loop-counter> from <x> to <y>

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

repeat with i from 1 to the count of the friends_list
   display dialog (item i of the friends_list)
end repeat
```



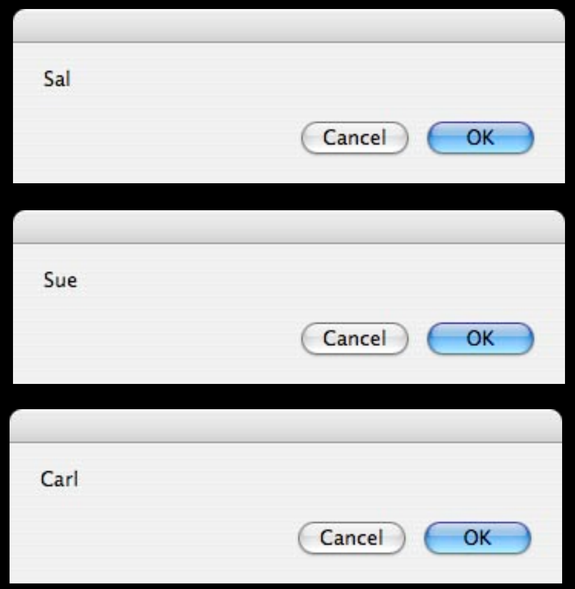
Repeat with <loop-counter> from <x> to <y>

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

repeat with i from 3 to the count of the friends_list
   display dialog (item i of the friends_list)
end repeat
```



Repeat with <loop-counter> from <x> to <y>

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

repeat with i from 1 to 3
   display dialog (item i of the friends_list)
end repeat
```

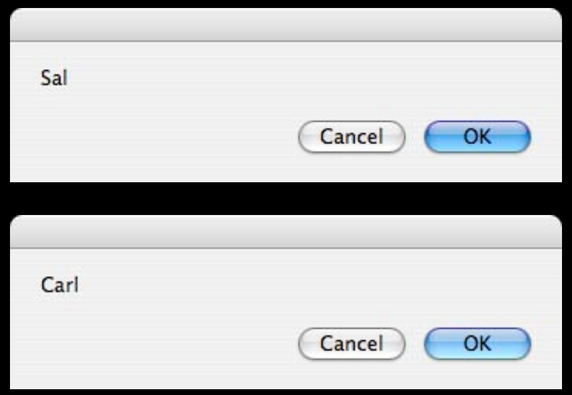


Repeat with <loop-counter> from <x> to <y>

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

repeat with i from 1 to the count of the friends_list by 2
   display dialog (item i of the friends_list)
end repeat
```

Repeat with <loop-counter> from <x> to <y>

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

repeat with i from 1 to the count of the friends_list by 2
   display dialog (item i of the friends_list)
end repeat
```

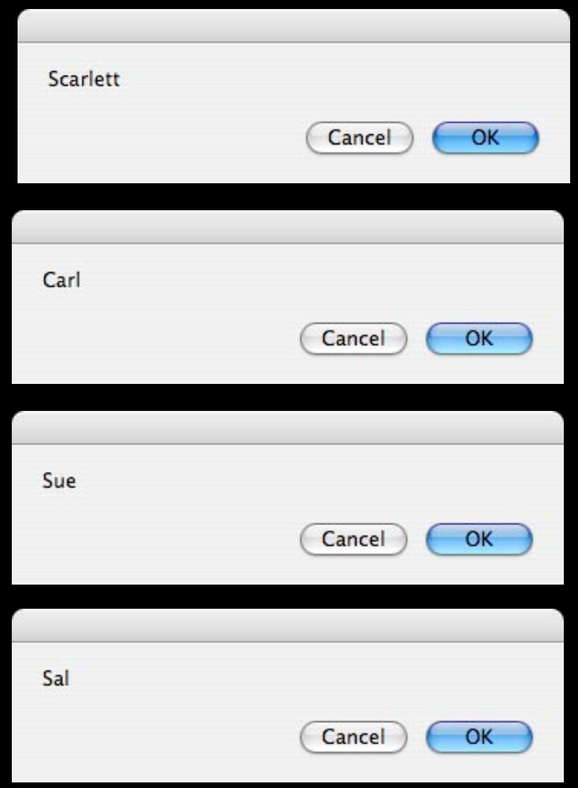


Repeat with <loop-counter> from <x> to <y>

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

repeat with i from the count of the friends_list to 1 by -1
   display dialog (item i of the friends_list)
end repeat
```

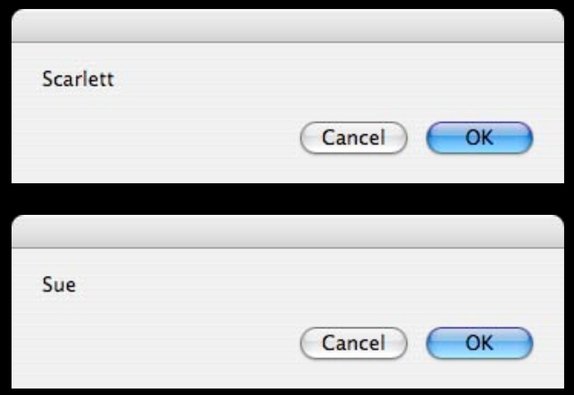
Repeat with <loop-counter> from <x> to <y>

```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

repeat with i from the count of the friends_list to 1 by -1
   display dialog (item i of the friends_list)
end repeat
```



Repeat with <loop-counter> from <x> to <y>

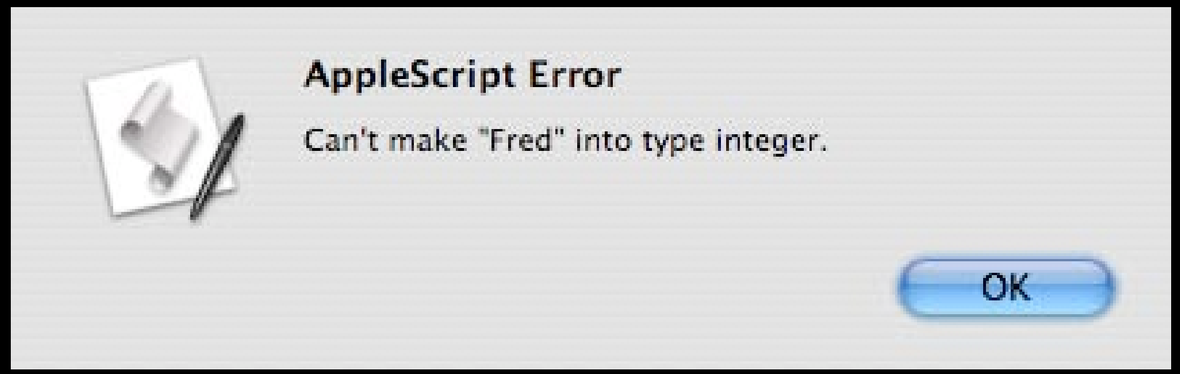
```
set the friends_list to {"Sal", "Sue", "Carl", "Scarlett"}

repeat with i from the count of the friends_list to 1 by -2
   display dialog (item i of the friends_list)
end repeat
```



Error Handlers

Error Handler

```
"Fred" as integer
```



Error Handler

```
try
   "Fred" as integer
end try
```

Error Handler

```
try
   "Fred" as integer
end try
```

Error Handler

```
try
   "Fred" as integer
end try
```

Error Handler

```
try
   "Fred" as integer
on error
   display dialog "There was a problem." buttons {"Cancel"} default button 1
end try
```

Error Handler

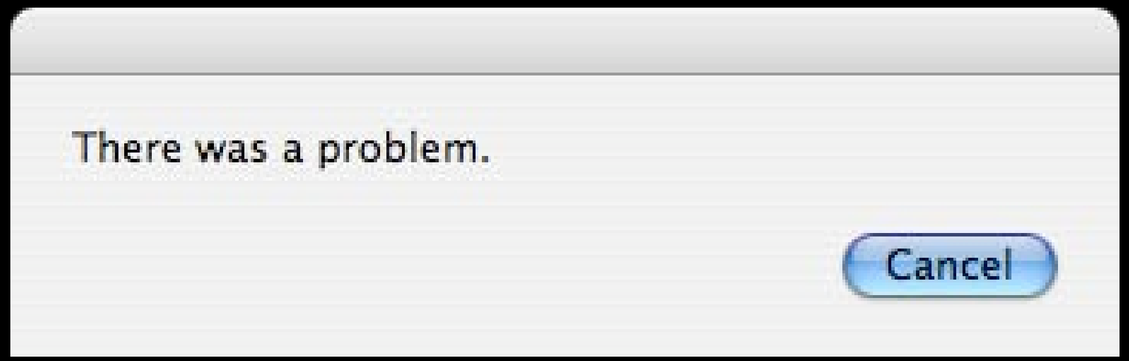
```
try
   "Fred" as integer
on error
   display dialog "There was a problem." buttons {"Cancel"} default button 1
end try
```

Error Handler

```
try
   "Fred" as integer
on error
   display dialog "There was a problem." buttons {"Cancel"} default button 1
end try
```

Error Handler

```
try
   "Fred" as integer
on error
   display dialog "There was a problem." buttons {"Cancel"} default button 1
end try
```



Error Handler

```
try
   "Fred" as integer
on error error_message
   display dialog error_message buttons {"Cancel"} default button 1
end try
```

Error Handler

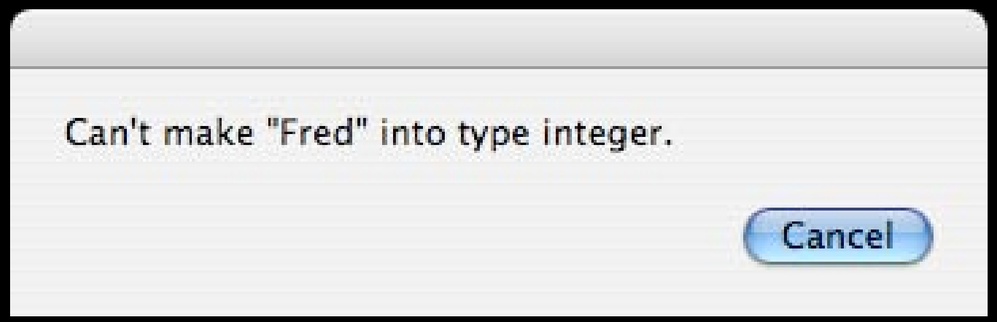
```
try
   "Fred" as integer
on error error_message
   display dialog error_message buttons {"Cancel"} default button 1
end try
```

Error Handler

```
try
   "Fred" as integer
on error error_message
   display dialog error_message buttons {"Cancel"} default button 1
end try
```

Error Handler

```
try
   "Fred" as integer
on error error_message
   display dialog error_message buttons {"Cancel"} default button 1
end try
```

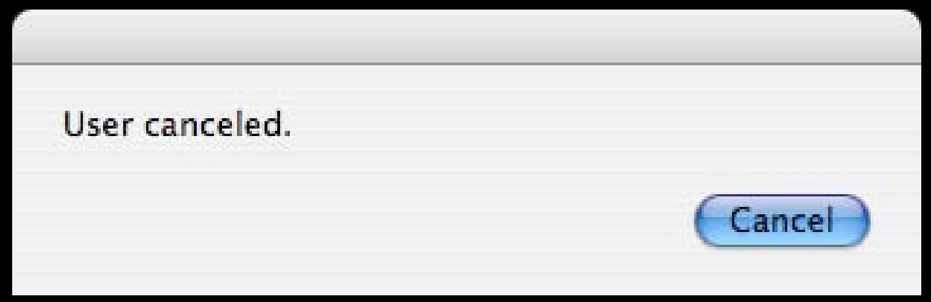


Error Handler

```
try
   set this_file to choose file of type "JPEG" without invisibles
   tell application "Image Events"
      launch
      set this_image to open this_file
      flip this_image with horizontal
      save this_image with icon
      close this_image
   end tell
on error error_message
   display dialog error_message buttons {"Cancel"} default button 1
end try
```

Error Handler

```
try
   set this_file to choose file of type "JPEG" without invisibles
   tell application "Image Events"
      launch
      set this_image to open this_file
      flip this_image with horizontal
      save this_image with icon
      close this_image
   end tell
on error error_message
   display dialog error_message buttons {"Cancel"} default button 1
end try
```



Error Handler

```
try
   set this_file to choose file of type "JPEG" without invisibles
   tell application "Image Events"
      launch
      set this_image to open this_file
      flip this_image with horizontal
      save this_image with icon
      close this_image
   end tell
on error error_message number error_number
   display dialog error_message buttons {"Cancel"} default button 1
end try
```

Error Handler

```
try
   set this_file to choose file of type "JPEG" without invisibles
   tell application "Image Events"
      launch
      set this_image to open this_file
      flip this_image with horizontal
      save this_image with icon
      close this_image
   end tell
on error error_message number error_number
   display dialog error_message buttons {"Cancel"} default button 1
end try
```

Error Handler

```
try
   set this_file to choose file of type "JPEG" without invisibles
   tell application "Image Events"
      launch
      set this_image to open this_file
      flip this_image with horizontal
      save this_image with icon
      close this_image
   end tell
on error error_message number error_number
   display dialog error_message buttons {"Cancel"} default button 1
end try
```

Error Handler

```
try
   set this_file to choose file of type "JPEG" without invisibles
   tell application "Image Events"
      launch
      set this_image to open this_file
      flip this_image with horizontal
      save this_image with icon
      close this_image
   end tell
on error error_message number error_number
   if the error_number is not -128 then
      display dialog error_message buttons {"Cancel"} default button 1
   end if
end try
```

Error Handler

```
try
   set this_file to choose file of type "JPEG" without invisibles
   tell application "Image Events"
      launch
      set this_image to open this_file
      flip this_image with horizontal
      save this_image with icon
      close this_image
   end tell
on error error_message number error_number
   if the error_number is not -128 then
      display dialog error_message buttons {"Cancel"} default button 1
   end if
end try
```

Sub-routines and Handlers

Sub-routine

```
say "Hello, Bob. How are you?" using "Vicki"
delay 1
say "Great! What’s new with you?" using "Bruce"
delay 1
say "Not much with me, but Apple is sure kicking posterior!" using "Vicki"
delay 1
say "You got that right!" using "Bruce"
```

Sub-routine

```
speak("Hello, Bob. How are you?", 1)
speak("Great! What's new with you?", 2)
speak("Not much with me but Apple is sure kicking posterior!", 1)
speak("Word up!", 2)

on speak(this_text, voice_indicator)
   if the voice_indicator is 1 then
      say this_text using "Vicki"
   else
      say this_text using "Bruce"
   end if
   delay 1
end speak
```

Sub-routine

```
speak("Hello, Bob. How are you?", 1)
speak("Great! What's new with you?", 2)
speak("Not much with me but Apple is sure kicking posterior!", 1)
speak("Word up!", 2)

on speak(this_text, voice_indicator)
   if the voice_indicator is 1 then
      say this_text using "Vicki"
   else
      say this_text using "Bruce"
   end if
   delay 1
end speak
```

Sub-routine

```
speak("Hello, Bob. How are you?", 1)
speak("Great! What's new with you?", 2)
speak("Not much with me but Apple is sure kicking posterior!", 1)
speak("Word up!", 2)

on speak(this_text, voice_indicator)
   if the voice_indicator is 1 then
      say this_text using "Vicki"
   else
      say this_text using "Bruce"
   end if
   delay 1
end speak
```

Sub-routine

```
speak("Hello, Bob. How are you?", 1)
speak("Great! What's new with you?", 2)
speak("Not much with me but Apple is sure kicking posterior!", 1)
speak("Word up!", 2)

on speak(this_text, voice_indicator)
   if the voice_indicator is 1 then
      say this_text using "Vicki"
   else
      say this_text using "Bruce"
   end if
   delay 1
end speak
```

Sub-routine

```
speak("Hello, Bob. How are you?", 1)
speak("Great! What's new with you?", 2)
speak("Not much with me but Apple is sure kicking posterior!", 1)
speak("Word up!", 2)

on speak(this_text, voice_indicator)
   if the voice_indicator is 1 then
      say this_text using "Vicki"
   else
      say this_text using "Bruce"
   end if
   delay 1
end speak
```

Sub-routine

```
speak("Hello, Bob. How are you?", 1)
speak("Great! What's new with you?", 2)
speak("Not much with me but Apple is sure kicking posterior!", 1)
speak("Word up!", 2)

on speak(this_text, voice_indicator)
   if the voice_indicator is 1 then
      say this_text using "Vicki"
   else
      say this_text using "Bruce"
   end if
   delay 1
end speak
```

Sub-routine

```
speak("Hello, Bob. How are you?", 1)
speak("Great! What's new with you?", 2)
speak("Not much with me but Apple is sure kicking posterior!", 1)
speak("Word up!", 2)

on speak(this_text, voice_indicator)
   if the voice_indicator is 1 then
      say this_text using "Vicki"
   else
      say this_text using "Bruce"
   end if
   delay 1
end speak
```

Sub-routine

```
tell application "Finder"
   activate
   close every window
   open the startup disk
   my speak("Hello, Bob. How are you?", 1)
   set the target of the front Finder window to home
   my speak("Great! What’s new with you?", 2)
   set the target of the front Finder window to the startup disk
   my speak("Not much with me but Apple is sure kicking posterior!", 1)
   set the target of the front Finder window to home
   my speak("Word up!", 2)
   close every window
end tell

on speak(this_text, voice_indicator)
   if the voice_indicator is 1 then
      say this_text using "Vicki"
   else
      say this_text using "Bruce"
   end if
   delay 1
end speak
```

Sub-routine

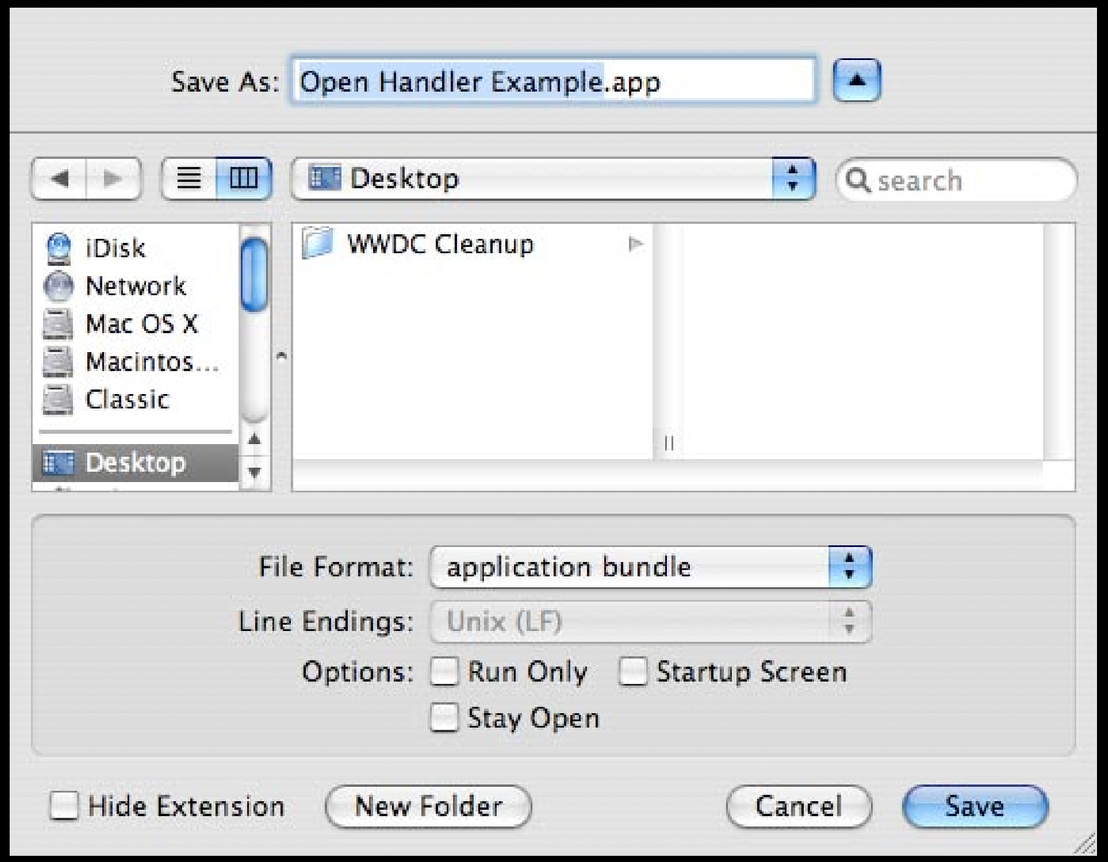
```
tell application "Finder"
   activate
   close every window
   open the startup disk
   my speak("Hello, Bob. How are you?", 1)
   set the target of the front Finder window to home
   my speak("Great! What’s new with you?", 2)
   set the target of the front Finder window to the startup disk
   my speak("Not much with me but Apple is sure kicking posterior!", 1)
   set the target of the front Finder window to home
   my speak("Word up!", 2)
   close every window
end tell

on speak(this_text, voice_indicator)
   if the voice_indicator is 1 then
      say this_text using "Vicki"
   else
      say this_text using "Bruce"
   end if
   delay 1
end speak
```

Sub-routine

```
tell application "Finder"
   activate
   close every window
   open the startup disk
   my speak("Hello, Bob. How are you?", 1)
   set the target of the front Finder window to home
   my speak("Great! What’s new with you?", 2)
   set the target of the front Finder window to the startup disk
   my speak("Not much with me but Apple is sure kicking posterior!", 1)
   set the target of the front Finder window to home
   my speak("Word up!", 2)
   close every window
end tell

on speak(this_text, voice_indicator)
   if the voice_indicator is 1 then
      say this_text using "Vicki"
   else
      say this_text using "Bruce"
   end if
   delay 1
end speak
```

Open Handler

```
on open these_items
   display dialog (count of these_items)
end open
```



Open Handler

```
property type_list : {"JPEG"}
property extension_list : {"jpg", "jpeg"}

on open these_items
   repeat with i from 1 to the count of these_items
      set this_item to item i of these_items
      set the item_info to info for this_item
      if (folder of the item_info is false) and (alias of the item_info is false)
            and ((the file type of the item_info is in the type_list) or (the name
            extension of the item_info is in the extension_list)) then
         process_item(this_item)
      end if
   end repeat
end open

on process_item(this_item)
   -- NOTE that the variable this_item is a file reference in alias format
   -- FILE PROCESSING STATEMENTS GOES HERE
end process_item
```

Open Handler

```
property type_list : {"JPEG"}
property extension_list : {"jpg", "jpeg"}

on open these_items
   repeat with i from 1 to the count of these_items
      set this_item to item i of these_items
      set the item_info to info for this_item
      if (folder of the item_info is false) and (alias of the item_info is false)
            and ((the file type of the item_info is in the type_list) or (the name
            extension of the item_info is in the extension_list)) then
         process_item(this_item)
      end if
   end repeat
end open

on process_item(this_item)
   -- NOTE that the variable this_item is a file reference in alias format
   -- FILE PROCESSING STATEMENTS GOES HERE
end process_item
```

Open Handler

```
property type_list : {"JPEG"}
property extension_list : {"jpg", "jpeg"}

on open these_items
   repeat with i from 1 to the count of these_items
      set this_item to item i of these_items
      set the item_info to info for this_item
      if (folder of the item_info is false) and (alias of the item_info is false)
            and ((the file type of the item_info is in the type_list) or (the name
            extension of the item_info is in the extension_list)) then
         process_item(this_item)
      end if
   end repeat
end open

on process_item(this_item)
   -- NOTE that the variable this_item is a file reference in alias format
   -- FILE PROCESSING STATEMENTS GOES HERE
end process_item
```

Open Handler

```
property type_list : {"JPEG"}
property extension_list : {"jpg", "jpeg"}

on open these_items
   repeat with i from 1 to the count of these_items
      set this_item to item i of these_items
      set the item_info to info for this_item
      if (folder of the item_info is false) and (alias of the item_info is false)
            and ((the file type of the item_info is in the type_list) or (the name
            extension of the item_info is in the extension_list)) then
         process_item(this_item)
      end if
   end repeat
end open

on process_item(this_item)
   -- NOTE that the variable this_item is a file reference in alias format
   -- FILE PROCESSING STATEMENTS GOES HERE
end process_item
```

Open Handler

```
property type_list : {"JPEG"}
property extension_list : {"jpg", "jpeg"}

on open these_items
   repeat with i from 1 to the count of these_items
      set this_item to item i of these_items
      set the item_info to info for this_item
      if (folder of the item_info is false) and (alias of the item_info is false)
            and ((the file type of the item_info is in the type_list) or (the name
            extension of the item_info is in the extension_list)) then
         process_item(this_item)
      end if
   end repeat
end open

on process_item(this_item)
   -- NOTE that the variable this_item is a file reference in alias format
   -- FILE PROCESSING STATEMENTS GOES HERE
end process_item
```

Open Handler

```
property type_list : {"JPEG"}
property extension_list : {"jpg", "jpeg"}

on open these_items
   repeat with i from 1 to the count of these_items
      set this_item to item i of these_items
      set the item_info to info for this_item
      if (folder of the item_info is false) and (alias of the item_info is false)
            and ((the file type of the item_info is in the type_list) or (the name
            extension of the item_info is in the extension_list)) then
         process_item(this_item)
      end if
   end repeat
end open

on process_item(this_item)
   -- NOTE that the variable this_item is a file reference in alias format
   -- FILE PROCESSING STATEMENTS GOES HERE
end process_item
```

Open Handler

```
property type_list : {"JPEG"}
property extension_list : {"jpg", "jpeg"}

on open these_items
   repeat with i from 1 to the count of these_items
      set this_item to item i of these_items
      set the item_info to info for this_item
      if (folder of the item_info is false) and (alias of the item_info is false)
            and ((the file type of the item_info is in the type_list) or (the name
            extension of the item_info is in the extension_list)) then
         process_item(this_item)
      end if
   end repeat
end open

on process_item(this_item)
   -- NOTE that the variable this_item is a file reference in alias format
   -- FILE PROCESSING STATEMENTS GOES HERE
end process_item
```

Open Handler

```
property type_list : {"JPEG"}
property extension_list : {"jpg", "jpeg"}

on open these_items
   repeat with i from 1 to the count of these_items
      set this_item to item i of these_items
      set the item_info to info for this_item
      if (folder of the item_info is false) and (alias of the item_info is false)
            and ((the file type of the item_info is in the type_list) or (the name
            extension of the item_info is in the extension_list)) then
         process_item(this_item)
      end if
   end repeat
end open

on process_item(this_item)
   -- NOTE that the variable this_item is a file reference in alias format
   -- FILE PROCESSING STATEMENTS GOES HERE
end process_item
```

Working with Lists

Tips for working with lists

http://developer.apple.com/wwdc2005/

Getting an item in a list

```
set the friends_list to {"Sal", "Sue", "Carl", "Bob", "Wanda", "Walter", "Zeke",
   "Scarlett", "Rhett"}

get the first item of the friends_list
   --> "Sal"
get item 1 of the friends_list
   --> "Sal"
get the last item of the friends_list
   --> "Rhett"
get item -1 of the friends_list
   --> "Rhett"
get the middle item of the friends_list
   --> ""Wanda"
get some item of the friends_list
   --> random result
```

Getting a range of items in a list

```
set the friends_list to {"Sal", "Sue", "Carl", "Bob", "Wanda", "Walter", "Zeke",
   "Scarlett", "Rhett"}

get items 3 thru 5 of the friends_list
   --> {"Carl", "Bob", "Wanda"}

get items 3 thru -1 of the friends_list
   --> {"Carl", "Bob", "Wanda", "Walter", "Zeke", "Scarlett", "Rhett"}

get the rest of the friends_list
   --> {"Sue", "Carl", "Bob", "Wanda", "Walter", "Zeke", "Scarlett", "Rhett"}
```

Comparison Operators

```
set the friends_list to {"Sal", "Sue", "Carl", "Bob", "Wanda", "Walter", "Zeke",
   "Scarlett", "Rhett"}

the friends_list contains "Sal"
   --> true

"Marge" is in the friends_list
   --> false
```

Setting an item in a list

```
set the friends_list to {"Sal", "Sue", "Carl", "Bob", "Wanda", "Walter", "Zeke",
   "Scarlett", "Rhett"}

set item 3 of the friends_list to "Sally"
get the friends_list
   --> {"Sal", "Sue", "Sally", "Bob", "Wanda", "Walter", "Zeke", "Scarlett",
   "Rhett"}
```

Adding items to a list

```
set the friends_list to {"Sal", "Sue", "Carl", "Bob", "Wanda", "Walter", "Zeke",
   "Scarlett", "Rhett"}

set the end of the friends_list to "Seal"
get the friends_list
   --> {"Sal", "Sue", "Sally", "Bob", "Wanda", "Walter", "Zeke", "Scarlett",
   "Rhett", "Seal"}

set the beginning of the friends_list to "Devon"
get the friends_list
   --> {"Devon", "Sal", "Sue", "Sally", "Bob", "Wanda", "Walter", "Zeke",
   "Scarlett", "Rhett", "Seal"}
```

Deleting items in a list

```
set the friends_list to {"Sal", "Sue", "Carl", "Bob", "Wanda", "Walter", "Zeke",
   "Scarlett", "Rhett"}

set part_A to items 1 thru 3 of the friends_list
set part_B to items 5 thru -1 of the friends_list
set the friends_list to part_A & part_B
get the friends_list
   --> {"Sal", "Sue", "Carl", "Wanda", "Walter", "Zeke", "Scarlett", "Rhett"}

-- removing the first item
set the friends_list to the rest of the friends_list
get the friends_list
   --> {"Sue", "Carl", "Bob", "Wanda", "Walter", "Zeke", "Scarlett", "Rhett"}
```

Reversing a list

```
set the friends_list to {"Sal", "Sue", "Carl", "Bob", "Wanda", "Walter", "Zeke",
   "Scarlett", "Rhett"}

set the friends_list to the reverse of the friends_list
get the friends_list
   --> {"Rhett", "Scarlett", "Zeke", "Walter", "Wanda", "Bob", "Carl", "Sue",
   "Sal"}
```

Iterating a list

```
set the friends_list to {"Sal", "Sue", "Carl", "Bob", "Wanda", "Walter", "Zeke",
   "Scarlett", "Rhett"}

repeat with i from 1 to the count of the friends_list
   set this_friend to item i of the friends_list
   display dialog this_friend
end repeat
```

Q&A

http://developer.apple.com/wwdc2005/

WWDC2005

Worldwide Developers Conference